A Hybrid QoS Management Scheme

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: Anne.Fladenmuller@masi.ibp.fr

Abstract

Management schemes for multimedia applications typically fall into two categories. Those which rely on central manager to reserve resources for each application and those that use application based adaptation to deliver a reasonable quality of service. The problems associated with these two approaches are discussed in this paper and a hybrid model of the two schemes proposed that mitigates many of these.

Keywords: Multimedia, Quality of Service Management, Adaptive Applications

1 Introduction

The requirements of multimedia applications can be defined in terms of their required Quality of Service (QoS). Currently there appears to be two approaches to managing the QoS of these applications. In the first approach before an application can run, it interacts with a QoS Manager which checks whether there are sufficient resources available to ensure a minimum QoS level. If there is enough resources, 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 schemes proposals 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 occur[1], the QoS manager needs to renegotiate the QoS levels, this can be an involved and

[1] this will be common in mobile environments

time consuming exercise. That 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 based on this principle have also been reported [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, graceful degradation and are robust. We believe that a hybrid architecture which supports centralised management of QoS coupled with application based adpativity can overcome the limitations of both the above schemes.

2 Different approaches for QoS management

2.1 management with or without resources reservation

Several approaches consider multimedia management either with or without QoS guarantee. QoS guarantee ensures users that applications are started only if a minimum quality can be provided. Quality thresholds are determined by applications programmers and transmitted to a QoS manager. A negotiation phase between applications and a QoS manager determines whether the specified quality threshold can be guaranteed. This negotiation phase starts with a mapping of the application's QoS specifications that define applications communication profiles. These profiles determine the amount of system and network resources corresponding to applications desirable or minimum QoS. The negotiation phase ends when it is ensured that there are sufficient resources available to run the application and after these resources have been reserved. So, the application has a guaranteed service quality. In this scheme the QoS management is done by the application for a QoS specification phase, as well as by the underlying system and network (reservation and usage monitoring of resources). If resource reservation can be done, such a scheme would work very well. However most current networks and operating systems do not provide resource reservation facilities.

Other available schemes attempt to match the applications QoS to their operating environment. These schemes are best suited to situations where resource reservation is not possible or limited. Since no resource reservation is performed, the QoS negotiation phase is simplier. The QoS specification and mapping, communication profile determination are no longer necessary. The management of QoS is then preformed by the applications themselves. In this second approach, the application programmer selects the rules for adaptation of their application, that is the services and the modifications to the ones that are required to deliver the best quality given the environment. No further control for QoS management by the underlying support system seems necessary. As applications programmers are the best persons to determine which services are the most appropriate for their applications and what quality will be expected by the users, this approach seems to be a very sensible one.

In this paper we are focusing on QoS management when no resource reservation is possible. Section 3 explains how an application can adapt its execution to match the system load and show the risks applications can face when taking blind decisions about their environment. In section 4, we will present a hybrid management framework and describe the implemented prototype.

3 Adaptive applications

3.1 Advantages of running adaptive applications

Adaptive applications basically change their execution pattern to suit the operating environment. The applications must be capable of detecting the changes in their environments. This can not be done at system level because the system would need to know every simple applications need. With an effectively infinite variety of applications this is not realistic. Classing the needs of each application by the media they use is also inappropriate. The video play-back and video editor example provided in [6] illustrate this clearly. The main requirement of the video player is to keep good synchronisation between voice and video, while the video editor expects precise editing and therefore needs a guarantee on every frame, because is to be seen by users. The expected degradation scheme should then be different depending on the application. Our conclusion is that a single degradation policy can not satisfy the requirements of diverse applications.

The approach adopted with adaptive applications has several advantages. As adaptive applications monitor themselves their quality, they can react dynamically to environments changes. This is particularly useful in non stable environments like the ones relying on mobile computer. Modifications of application execution pattern are defined in the application degradation/upgradation schemes. These schemes are determined by the application programmers, and match to applications requirements and users expectation. As application programmers are aware of their applications requirements, they are the ones that can define the most appropriate services and their minimum and maximum QoS.

For a degradation script, an application reduces its service to keep an acceptable QoS, for example it displays black and white movie rather than a colour one to get at least tolerable frame rate. The decrease of a service level reduces the amount of system or network resources used by the application. Thus, applications are also aware of their environment and do not try to overload the system to get the higher service. They adapt their service to match to the system or network load.

To run experiments about the execution of such applications, we have implemented two different samples of adaptive applications. In the next section we are going to describe the execution pattern of these two applications.

3.2 Some examples of adaptive application

The first adaptive application implemented to run our expirements is a simple prototype sensitive to system load modifications. For this program, five service levels have been defined, each corresponding to a program priority. The lowest (resp. highest) service level corresponds to the lowest (resp. highest) program priority. This priority can be modified dynamically using the **nice** unix command, when a significant modification of the available system resources is detected.

In order to detect system load modifications the program executes the same function several times and measures the interval of time for each execution. As some other programs can run concurrently to ours on the system, the execution time of this function varies. The program compares the *i*-th execution of the function E_{i-1} with the (*i-1*)-th E_{i-1} and detects increase or decrease of execution times. If another program uses more resources, the execution time of our function increases $E_i > E_{i-1}$. Then, the model measures a decrease in available system resources and it decreases to a lower service level by reducing the program priority. For our simulations, we have considered execution time modifications as significant if they reach 10 percent of the previous execution time.

The second application is an adaptive Mpeg-play. This application modifies the video display to keep a reasonable frame rate when dynamic system load modifications occur. Thus, significant modifications change the service level of the application. For adaptive Mpeg-play, each of the 3 defined service levels corresponds to a given video size and colour. The highest service level corresponds to a big format colour video, the middle one to a small format of color video and the lowest one refers to a small format of black & white video.

The program averages the display time of n mpeg-play pictures A_n to cope with the difference of time needed for coding the different types of frames. To each frame i (with $i > n-1$) corresponds an average display time A_{ni}. This average display rate is expected to stay in a range $A_{MAX} > A_n > A_{MIN}$. When A_{ni} becomes greater than A_{MAX} the program detects some new resources are available and the service level is increased when it is not the maximum one already. If A_{ni} becomes lower than A_{MIN} the program considers the quality is not sufficient, so the service level is decreased when it is not the lower one.

For the experiments we set n at 10, the maximum averaged frame rate at 20 images per second and the minimum averaged frame rate to 5.

3.3 Experiments

We were interested in studying the behaviour of adaptive applications running concurrently on the same system. Therefore we have run concurrently our two adaptive applications and compare the service level of both [14] application. The experiments show it is not possible to predict the evolution of adaptive applications. Moreover when there is a lack of resources, it is possible for one adaptive application to decrease its service level whereas another running concurrently would improve the service given to its users. Thus one application could benefit from the degradation of another.

As a conclusion of these experiments, we constate that without a system level management of some form, adaptive applications can unfairly distribute the availabale resources. To avoid these problems, we present in the next section a hybrid management model.

4 A hybrid model for applications management

The model has two components, the QoS supervisor and the admission control entity. The supervisor prevents applications from taking "blind" decisions to modify their execution.

In the case of system load increasing, where the applications adaption policy has to keep a good quality of service rather than a good service of poor quality, the supervisor will tend to make all running adpative applications converge to a common service level. We intend to prove in the next section that the creation of a supervisor reduces problems caused by non-deterministic behaviour of such applications and unfair distribution of resources among adaptive applications.

Nevertheless, the supervisor does not guarantee that once an application is started, it can be run at its minimum service with reasonable quality. When there is not enough resources to run all of the adaptive applications at the desired service level, all applications will converge to a degraded service level. When the amount of resources is not sufficient to continue working correctly at its lowest service level, the application degradation policy might decide to "auto-kill" itself or will continue running with a very poor quality. If several adaptive applications are running concurrently, it seems unfair to have one chosen arbitrarilly to exit from its execution, or have all applications running in low quality with degraded mode. Such a situation can easily arise on current multi-user systems, since they allow users to start new applications whenever they want, in an environment with no resource reservation (hypothesis for the framework).

To compensate this other unfair situations, we have also defined an admission control mechanism to control the number of adaptive applications running on the system. So, before an application can commence, the admission controler has to determine if it will unacceptably degrade the quality of currently working adaptive applications. The admission control mecanism is described in section 4.2.

An overview of the structure of this system is given in Figure 1.

4.1 Adaptive application supervisor

This section explains the functionalities of the supervisor and analyse results of the experiments done with this adaptive application supervisor.

4.2 Admission control mechanism

We present the admission control model we are currently implementing. The choice of using management information bases (MIBs) to keep information about system resources utilisation and adaptive applications needs, will be explained. Then, we will describe the

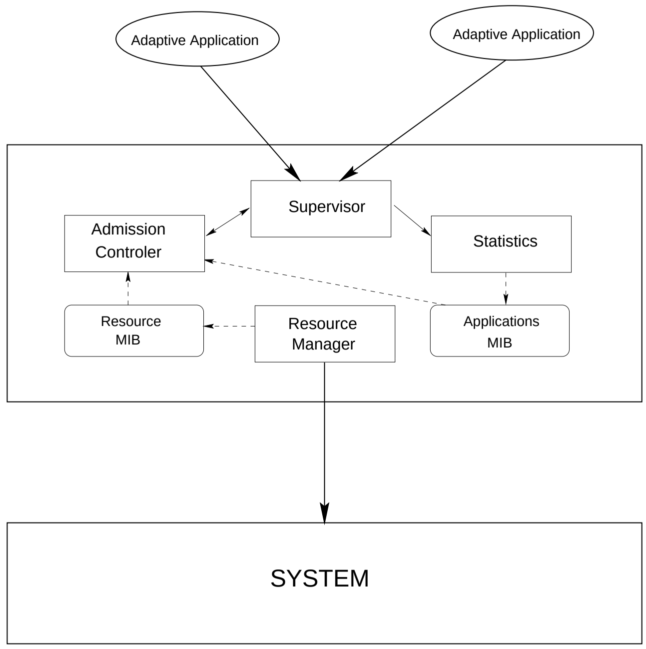

Figure 1: Implementation scheme for an end-to-end management

admission control implementation model with its diverse components: the entity responsible for making statistics about applications resources usage, the SNMP agent in charge of MIBs management, and the one responsible for admission control test.

5 Conclusion

In some environments where no resources reservation can be done, a good scheme consists of giving applications the possibility of modifying their quality to make the best use of the available resources. Therefore we have explained in this paper the necessity of these adaptive applications for such environments.

Nevertheless, running such adaptive applications leads to unfair resources repartitioning. 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. We have written a *AQ supervisor* and have had some good results for the management of different adaptive applications.

To provide applications with a minimum quality, it seems reasonable to limit the number of demanding applications. Therefore we have added in our model an admission control mechanism. We have also decribed in this paper the implementation model for the admission control and we intend to study the cost of adding such a component to our future experiments.

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", Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, december 1994.

[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, Boston, MA, 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", High Performance Computing Conference, Singapore, September 1994.

[10] A. Seneviratne and H. S. Cho, “Quality of Service Mapping in Distributed Multimedia Systems” IEEE MMNet’95, Aizu, Japan, September 1995.

[11] J-C Bolot, T. Turletti, “A rate control mechanism for packet video in the Internet”, IEEE INFOCOM’94, June 1994.

[12] A. Campbell, G. Coulson, F. Garcia, D. Hutchinson, and H. Leopold, “Integrated Quality of Service for Multimedia Communications”, IEEE INFOCOM’93, 1993.

[13] 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.

[14] A. Fladenmuller, A.Seneviratne, E. Horlait, ”A Hybrid QoS Management Scheme for Distributed”, Second workshop on PROMS, Salzburg, 1995.

[15] 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.

[16] M. Rose, “RFC 1418 SNMP over OSI”, March 1993.