TR-AC-0036 003

適応的QoS制御における パーソナルエージェントの提案

A Personal Agent for a QoS Management System

Antoine RAUX　中岡 謙

1999.12.24

ATR環境適応通信研究所

適応的QoS制御におけるパーソナルエージェントの提案
A Personal Agent for a QoS Management System

（株）エイ・ティ・アール環境適応通信研究所
第一研究室
Antoine RAUX　　中岡 謙

ATR Adaptive Communications Research Laboratories
Department 1
Antoine RAUX　　Ken NAKAOKA

あらまし

分散型マルチメディアアプリケーションにおいて、ユーザの要求を充足すべくネットワークや端末などの限られた資源を効率的に活用するためには、ユーザの要求や利用可能な資源に基づいてメディア品質を適応的に調整する必要がある。本研究室では、マルチエージェントによる直接的あるいは間接的な連携機能を利用した適応的 QoS 制御方式の研究を行っており、最上位層に位置するパーソナルエージェントは、ユーザとシステムとの間におけるインタラクションを通してユーザの要求・嗜好を獲得し、ユーザのメディアに対する好みを反映したユーザ QoS 調整を行う。

本レポートは、本研究室で進めている適応的 QoS 制御方式の研究の一環として、実験システム上にパーソナルエージェントを実装するという実習テーマのもと、学外実習生である Antoine RAUX による 1999 年 4 月～7 月の実習成果をまとめたものである。

Résumé

Afin d'assurer des services en ligne de la meilleure qualité possible et de différencier ces services en fonction des besoins des particuliers et des entreprises, la notion de Qualité de Service (Quality of Service, QoS), déjà présente sur certains réseaux à haut débit comme ATM, est en passe de se généraliser à l'Internet grâce à de nouveaux protocoles de réservation de ressources comme RSVP. Pour que l'utilisateur puisse tirer le meilleur parti des ressources limitées dont il dispose, il est nécessaire que les applications multimédia utilisant les réseaux adaptent la qualité de leur prestation en fonction des choix de l'utilisateur et des ressources disponibles. Le projet lié à ce stage d'option scientifique visait à concevoir et développer un agent, intégré dans un système multi-agent de gestion de la QoS, prenant en charge l'acquisition des préférences de l'utilisateur de manière à s'adapter au mieux aux conditions matérielles. Trois modules composent cet agent: l'interface graphique traditionnelle, le module d'apprentissage comportant des fonctions élémentaires d'intelligence artificielle et le module de communication et de traduction des choix de l'utilisateur en paramètres liés aux applications multimédia. L'implémentation en Java se limite au cas d'une application diffusant de la vidéo à travers un réseau local, simulant des services de téléconférence ou de vidéo à la demande. A l'heure de la rédaction de ce document, seuls les premiers tests ont pu être effectués. Si l'accessibilité est à améliorer avant de pouvoir utiliser le Personal Agent dans des conditions réelles, il semble constituer un bon outil expérimental. Des expériences et des améliorations sont donc proposées en fin de rapport.

Abstract

In order to provide the best possible on-line services according to the needs of individuals and companies, the concept of Quality of Service (QoS) has become more and more important. Formerly provided on high-speed networks such as ATM, the QoS concept has become part of the Internet thanks to new resource reservation protocols like RSVP. To allow users to make the most out of the limited resources at their disposal, it is necessary that networked-based multimedia applications adapt their quality according to users' choices and available resources. As part of ACR Labs' researches on a Multi-Agent QoS Management System, this project aimed at conceiving and developing an agent in charge of the acquisition of user's preferences. It guides the adaptation to hardware conditions according to those preferences. Three modules compose this agent: the traditional Graphical User Interface, the Learning Module with some simple Artificial Intelligence functions, and the Communication Module also in charge of the translation of user's choices to application settings. The Java implementation, simulating teleconference and video-on-demand services, only considers a local-network-based video application. Only the first tests could be held so far. Although the agent's accessibility must be improved before using it in real conditions, it seems to be a good experimental tool. Experiments and improvements are thus proposed at the end of the report.

TABLE OF CONTENTS

1 Introduction

1.1 The concept of Quality of Service (QoS)

A fashionable buzzword: QoS

Due to its huge success and its worldwide coverage, the Internet is bound to evolve. At first a land of amateurs, crackers and surfers, it should become a real global secure network for companies and individuals. However, professionals will not use it for critical tasks as long as Internet Service Providers (ISP) cannot provide reliable services. The concept of Quality of Service, whose meaning is still not universally defined, is hence beginning to be used in a commercial way to evaluate and compare the ISP[1]. Internal networks such as ATM already provide QoS services, e.g. allow users to reserve bandwidth that is therefore guaranteed. The extension of the concept to the "best effort" Internet, although not so simple, is on the way. It is also a great challenge for the ISP who want to provide different classes of service, with different prices of course. Just like the "business class" in an airplane. In the future, different Internet users with different needs (and different amounts of money) will get different services.

Our definition of the QoS

More technically, we define the QoS as the description, at different levels, of a multimedia service delivered through a network[2]. This description is based on the fundamental parameters of multimedia elements, which can be sounds, video clips, raw data, etc. For instance, a video sequence can be described by the size and quality of the image and the number of images per second (frame rate). If there is sound, we also have to consider the sample rate, the number of bits per sample, etc.
However, these parameters are all at "application level". Indeed, a human user will rather use more directly understandable words to describe a situation or his/her feelings in front of a certain situation. From his/her point of view, a video may be smooth, big and clear or on the contrary jumpy, small and blurred. These words are also a representation of the QoS, at "user level".
At the other end of the process, low-level hardware and protocols are not interested either in the description of the video through its image size and quality, frame rate, etc. At this level, the QoS has to describe the amount of data to be transmitted and computed, regardless to the "meaning" of this data. A bandwidth of 1 kilobyte per second and a utilization of 20% of the CPU are an example of the representation of the QoS at "resource level".

1.2 About adaptivity

A good definition and evaluation of the QoS would not be useful if the multimedia systems were not able to behave according to it. The aim of a QoS management system is to react to changes in external conditions affecting the server, the network or the client. This implies dealing with the three levels described above. The QoS management system must monitor the available resources at "resource level"; it must take into account user's preferences de-

scribed at “user level” and it must tell the client and server applications which settings must be done, at “application level”.
At “resource level”, to be adaptive means to change one’s behavior according to the amount of available resources.
At “user level”, to be adaptive means to behave according to user’s tastes.

1.3 A Multi-Agent System

Our solution to the QoS management problem is built on a multi-agent architecture. In the first place, we will try to give a definition of this concept and then describes how we designed it to fit our needs.

What is a Multi-Agent System?

The general concepts about Multi-Agent Systems described here are taken from Gerhard Weiβ’s work[3].
When speaking of Multi-Agent Systems, we must first give a definition of an agent. Although today many different definitions of what an agent is exist, it is generally admitted that being an agent implies having the following skills:

- Perception and reaction to its environment.
- Communication.
- Autonomy.

Basically, an agent is a “module” (that is a piece of program, a robot, etc.) that receives information from its environment, performs actions according to it and emits information towards the other agents. The environment consists in all that is exterior to the agent, i.e. the other agents and what is outside the application.
Compared with traditional centralized systems, Multi-Agent applications offer several advantages such as:

- Parallelism.
- Perception of complex environments (for example composed at the same time of a human user and a computer network).
- Modularity (you do not have to modify and recompile the whole application when you want to change the behavior of just one agent).
- Reusability.

Usually, Multi-Agent Systems also provide collective or single-agent learning, making an extensive use of artificial intelligence.

The layered agents

The Multi-Agent concept fits particularly well the needs of a QoS management system. Traditionally, the networks are thought of as the juxtaposition of different layers. Basically, we will consider three layers that correspond to the levels of QoS described in 1.1. Each agent is in charge of a certain layer of the system:

- The Stream Agents (SA) are at the lowest level and manage resource reservation and monitoring (CPU utilization, network bandwidth, etc).
- The Application Agent (AA) sends to the server application the requests in terms of QoS.
- The Personal Agent (PA) is at user’s level and is in charge of the interaction between the user and the system.

In addition to this, the AA and the PA are able to translate the QoS from their level to the lower one. The whole system is summarized in Fig. 1.

1.4 Purpose of the project

The system conceived and implemented by the group I of ACR Laboratories is built on the architecture described above. The purpose of this research project is the conception of the Personal Agent, and its integration in the existing context of Application and Stream Agents.
To validate our theoretical choices, we will program a Personal Agent. This implementation will be specialized in the QoS of video images. The tests will be performed with a video application, simulating teleconference as well as video-on-demand service. To make it simpler, we will not consider the "sound" aspect of the media and only deal with video images.

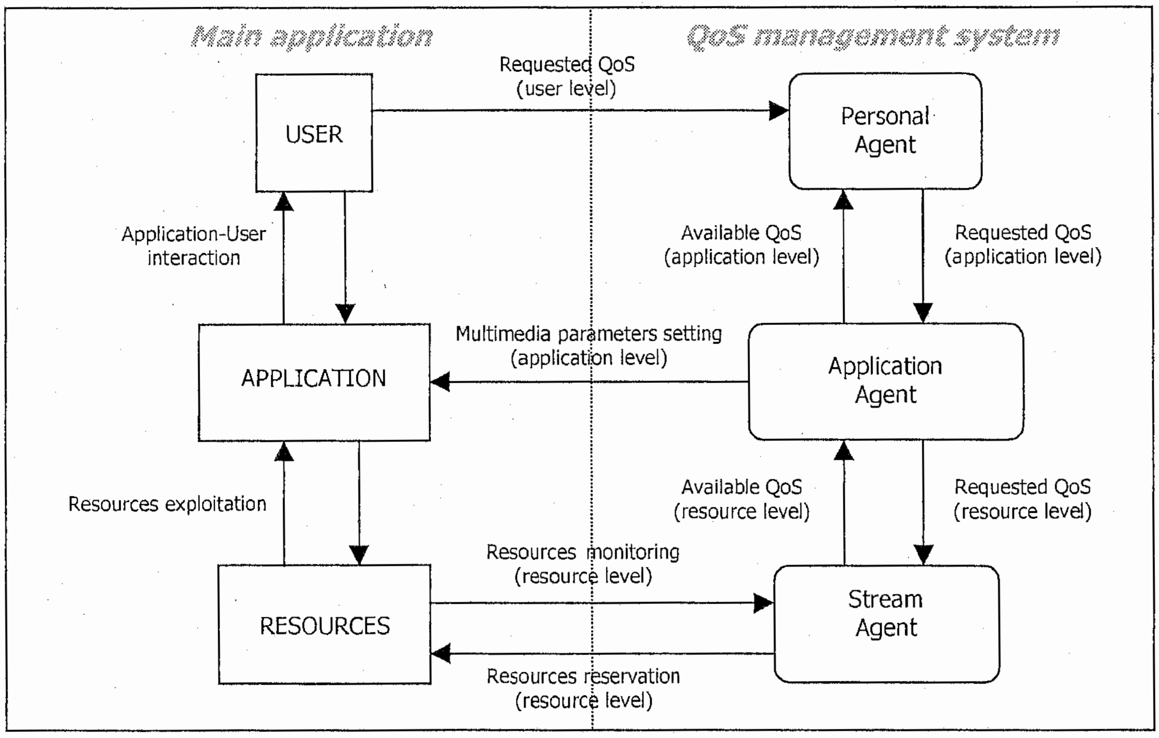
Main application
QoS management system
USER
Requested QoS (user level)
Personal Agent
Application-User interaction
Available QoS (application level)
Requested QoS (application level)
APPLICATION
Multimedia parameters setting (application level)
Application Agent
Resources exploitation
Available QoS (resource level)
Requested QoS (resource level)
Resources monitoring (resource level)
RESOURCES
Stream Agent
Resources reservation (resource level)

Fig. 1. The multi-agent architecture

2 Basic principles of the Personal Agent (PA)

2.1 Adaptation versus Universality

As seen before, the communication with the hardware is in the hands of the SA. The AA deals with the software as far as QoS is concerned. The role of the PA is at the upper level, the interaction with the user.

To simplify user's task as much as possible, two opposite approaches are commonly proposed:

- Setting through a single parameter
- Learning user's preferences

The first possibility, as investigated by Glynn Rogers et al.[4], is based on the assumption that we can find a *universal* invertible function to link a *cost* and the corresponding set of application parameters. The cost can be expressed in terms of reserved bandwidth for example. To find this invertible function, it is necessary to know the function *s* giving the value of user's satisfaction with respect to any set of QoS parameters. Then at a given cost, it will be possible to determine the maximum of *s*. Nevertheless, finding the function *s* will need researches in perception and subjective assessments and is not achieved yet.

The second approach, which we have chosen, does not make the assumption of universality. Indeed, we consider that the settings corresponding to one cost depend on the user. Therefore our PA must not only adapt to the *cost* but also to user's preferences. In this case, the role of the PA is to match user's tastes with the resource level Quality of Service offered by the network. Our PA must therefore use Machine Learning methods to acquire these preferences and to choose user's favorite settings.

2.2 Roles of the Personal Agent

First, the user, at least if he/she is not novice, may want to set the QoS entirely by him/herself. This is simply done through an appropriate "traditional" graphical user interface (GUI). Nevertheless, this is often fastidious or even impossible if the user is not familiar with technical terms such as frame rate or quantization factor (for JPEG images for example). Moreover, the user may not want to have to deal with preferences dialog boxes while watching a multimedia presentation. Therefore, the PA must cover two main tasks:

- Facilitating the understanding of applications parameters.
- Anticipating user's choices.

The first goal can be fulfilled through an appropriate translation between user level parameters and application parameters, and a good graphical interface.

The second one will require intervention from the user but we have to make it as simple and seldom as possible. Once user's profile has been designed (even if it can be modified afterwards if the user changes his/her mind), the PA will be able to determine by itself a QoS, translate it into Application level QoS (see 1.1) and send it to the AA.

2.3 The internal structure of the PA

We decided to program the PA in Java, using the Visual Café environment, because of its high portability, its advanced GUI components and its inter and intra application communication features. The communication was realized using method calls as far as the PA and the AA were concerned.

It seems natural to implement the functions seen above (see 2.2) into three different modules (see Fig. 2).

- The GUI manages the graphical interaction with the user.
- The Learning Module owns the capacity of learning user's preferences and using this knowledge.
- The Communication Module is in charge of the translation of the QoS and of the communication between the modules inside the PA and between the PA and the AA.

When first looking at this structure, several comments can be made.
The Communication Module has a central role, due to its function and is partly platform-dependent because of its link with the AA.
The Learning Module communicates only with the Com Module. It "learns" through this single communication. It can therefore be highly hardware-independent.
Finally, there are two critical "bridges" towards the exterior: user interaction through the GUI and hardware and software interaction through the Application Agent.

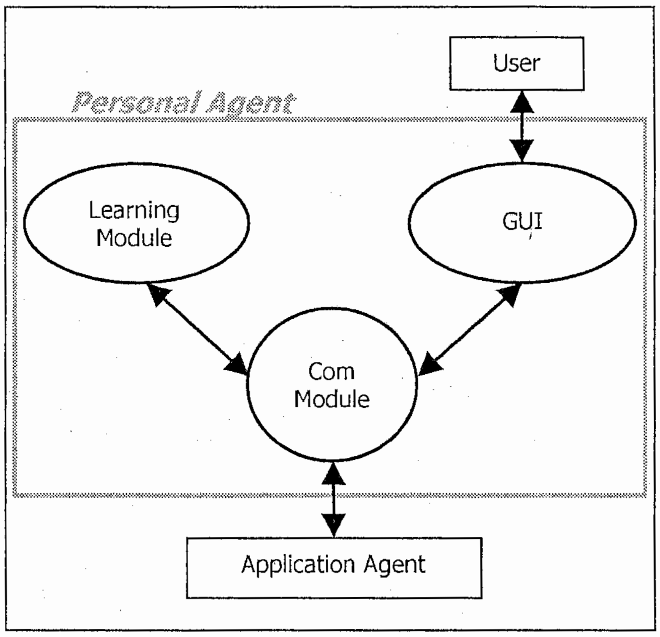

Fig. 2. *Internal structure of the Personal Agent*

2.4 The representation of the QoS

A Java class with three main fields represents the QoS at "user level" (UserVideoQoS, inheriting from UserMediaQoS, the abstract class parent of all the "user level" QoS description classes). These are the image size, the frame rate and the quality of the image. These parameters are the same as those used at "application level" but their values are different. At "user level", each parameter has a value between 0 and 100 expressing how good or bad it is. Moreover, size is defined by only one value (and not by width and height) as the proportions of the images are not supposed to be changed. To translate these values into application QoS, we use a reference QoS which consists in an "application level" QoS corresponding to the value 100. Every value inferior to 100 is translated proportionally to the "Best QoS". In fact, "user level" QoS values are percentages of the reference QoS.
In order to represent the state of the system in terms of resources, we have created the SituationDescriptor class. At present, the fields of this class are two integers: CPU utilization and network bandwidth. One method of the class allows getting one single real value representing the resources, calculated using CPU utilization and bandwidth. It is also possible to consider them separately. The addition of other factors such as delay, jitter, etc would not be difficult thanks to the encapsulation of the data.
We will now see how each module has been conceived and implemented and how the communication problems inside and outside the PA was resolved.

3 The GUI

3.1 Objectives

Through this interface, the user must be able to:

- See the state of the network (available resources).
- See the QoS currently active (whether it had been chosen directly by him or by the PA).
- Set the QoS with a good precision and inform the PA of these settings.

Most of all, it must be easy to understand and to use for an inexperienced user.

3.2 Implementation

The following choices have been made in order to build the interface. The class representing it is called VideoPABox and inherits from PABox (an abstract class which is not dedicated to video applications).
To be easily understandable by the user, we designed the interface with usual elements such as "sliders" and "progress bars". Progress bars provide a good visual representation of the situation and sliders an easy way to express user's preferences. These elements are set to represent percentages so they also fit well the definition of "user level" QoS, as a percentage of the reference QoS. A progress bar corresponds to each parameter, showing its current value. A slider allows the user to change this value. Being superposed and using the same scale, these two elements can be used together and their value easily compared by the user.
One more progress bar shows the utilization of the network. It is assumed that the AA knows how to evaluate the resources needed by one specific QoS and how to monitor the current available resources. At any time when one of these two values changes, the Com Module informs the GUI by calling one method of the PABox class (SetResourceMonitor). The progress bar shows the ratio between these two values: when the ratio is below one, the network is underused, when the ratio is above one, the network cannot afford the requested QoS and a value of one represents the ideal case when all the available resources are used.
When the user is not satisfied by the QoS automatically set, he/she can change the value of the parameters and then push the "Set" button. The GUI sends it to the Com Module by calling one of its methods (SetUserQoS). If the network can afford the requested QoS, it is actually set and stored in the database, as one point of user's profile.

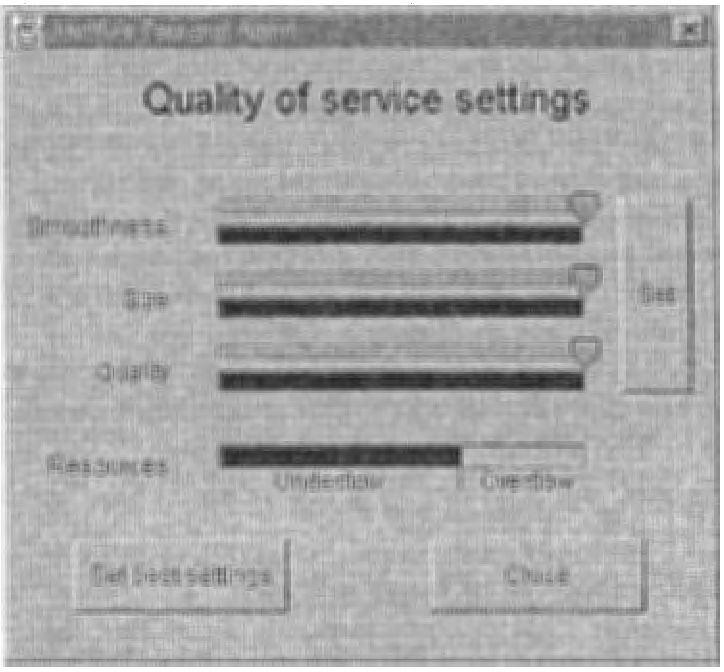

Fig. 3. *The GUI of the Personal Agent.*

A “Set best QoS” button has been added to allow experienced users to change directly the values of the “application level” “Best QoS”. Novice users should not use it, as it requires knowledge of the meaning and of the values of parameters such as image width and height, frame rate and quantization factor.

4 The Learning Module

4.1 Principles of Machine Learning applied to the PA

There are many different ways for intelligent systems to acquire knowledge[3]. Usual learning strategies cover from rote learning where all the knowledge is completely implanted from the beginning and no further transformation is required from the user, to learning by discovery where the agent gathers knowledge and skills by making observations, conducting experiments and generating and testing hypotheses.
From user’s point of view, the easiest way to “teach” its profile to the PA is to set the parameters as he/she wants them to be in certain conditions. That is why we have designed the PA to learn from examples. This means that it must change a general standard behavior into a user-specific behavior by acquiring information directly from the user. This information will be given at a few particular points (the examples). Intelligent learning resides in the ability of generalizing this information to previously unknown situations.
Another crucial aspect of a learning process is the learning feedback. This means how does the PA evaluate the utility of its proposition. In our case, utility means how good or how bad the user found the QoS proposed by the PA. Once again, there is a wide range of learning feedback methods. That is from supervised learning where the user should explicitly tell the PA which QoS is expected in a precise situation to unsupervised learning when the quality of the proposition is evaluated as a binary value (good/bad) and the system must learn through a trial/error process. As we have seen in paragraph 3, the PA uses a supervised learning method. Indeed, if the user was ever dissatisfied with the QoS selected by the PA, he/she can set it completely and so indicates exactly to the PA the correct response to a particular situation. This goes along with learning by examples.
The chosen multi-agent structure, in which each agent corresponds to a layer of the system implies that only the PA has learning capabilities, at least as far as user is concerned. Therefore no multi-agent learning algorithm is used. The PA’s learning process does not involve the other agents except as part of its environment.
To summarize what is written above, we can say that the PA’s learning process is:

- Example based.
- Supervised.
- Mono-agent.

4.2 User profiling process

To learn user’s preferences, we have chosen to determine his profile. In our project, the profile can be visualized as a set of graphs, one per “user level” multimedia parameter. Each graph represents the value of this parameter (between 0 and 100) with respect to a number representing the situation of the client-network-server system (hereinafter referred

to as the index). The index's value will be discussed in paragraph 4.3. The AA has to be able to evaluate the index of a given set of QoS parameters and calculate the index corresponding to the current situation, as monitored by the Stream Agents. The profile will be drawn by setting points (the examples seen in 4.1) and doing linear approximation between them.
At the very beginning, as we don't know anything about the user, three points are set. The first one corresponds to a value of 0 for each parameter and value of 0 for the index. The second point is associated to the best value of each parameter (100 at "user level") and to the corresponding index, which is evaluated using the AA. These two points make a straight line between the worst and the best network condition.

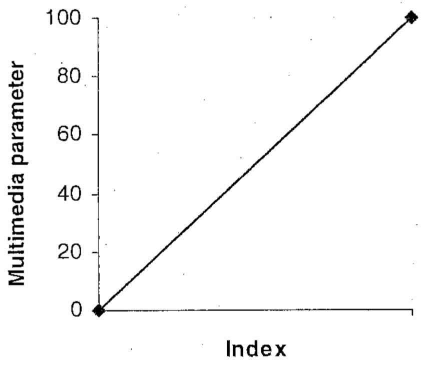

Fig. 4.a. *The initial shape of user's profile.*

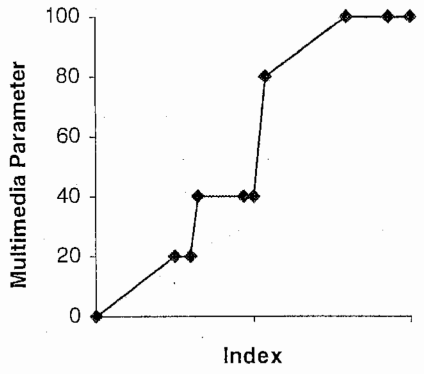

Fig. 4.b. *User's profile after the learning process. Diamonds represent the points actually set by the user.*

Then, anytime the user pushes the Set button, the AA calculates the index corresponding to this QoS and a point corresponding to the selected values of the QoS parameters is set in the graph. In order to prevent the system from being incoherent, a removal process had to be created. For example, the user could make a mistake and push the Set button although the current QoS did not correspond to his profile. If the index is near from a previously set point, this point is removed and replaced by the new one. This allows the profile to be quickly reshaped if user's tastes change or if he/she has made a mistake.

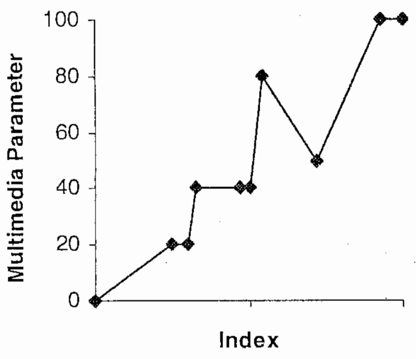

Fig. 5.a. *An incoherent point has been introduced in the profile.*

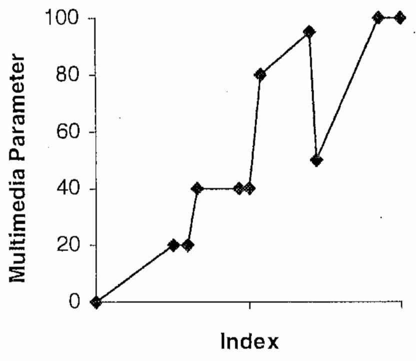

Fig. 5.b. *Without removal process, the incoherence remains.*

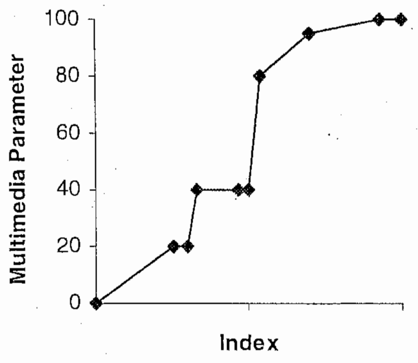

Fig. 5.c. *With a removal process, the incoherence is repaired after the first point is set.*

As said before, anytime the situation changes, the new QoS is determined by linear approximation between the previously set points.

4.3 The Learning Module index

At first we defined the index as the necessary bandwidth corresponding to a set of QoS parameters. As a first approximation, we neglected the other constraints such as CPU utilization, delay, jitter, etc. Nevertheless, there are still some drawbacks to using the raw bandwidth as an index. These drawbacks are linked with the non-linearity of the bandwidth with respect to user satisfaction. The dependence of the bandwidth on the QoS parameter is a complicated mathematical function that can be approximated using experimental measures of the bandwidth in different conditions. We chose the variables describing a QoS so that this function is crescent with any of them. An empiric property of the bandwidth is that it has quite an exponential behavior when the all the variables are strictly positive. For example in the case of the video sequence, the bigger the image is, the bigger the impact of an increase of the frame rate on the bandwidth will be. Each parameter having a positive influence on the bandwidth, a combined increase of these parameters will highly influence the value of the bandwidth.

The function describing user's satisfaction, yet depending on user, is more difficult to draw. Its mathematical definition is practically impossible to realize. Current researches in this domain imply joint works in psychology and computer science. The former researches of Glynn Rogers et al.[4] show that user's satisfaction may depend on each parameter as a sigmoid function, both low values and high values implying little variation in the satisfaction function. At a specific threshold, depending on the parameter and not precisely determined yet, the satisfaction increases quickly. For example below 2 images per seconds, the impression given by a video hardly changes (it remains "unsatisfactory"). Then between 5 and 20 images per second, it improves considerably (passing from "disagreeable" to "nearly smooth"). Finally, above 20 it does not change much (from "nearly smooth" to "smooth"). Another qualitative aspect of this function is that a good satisfaction level necessarily im-

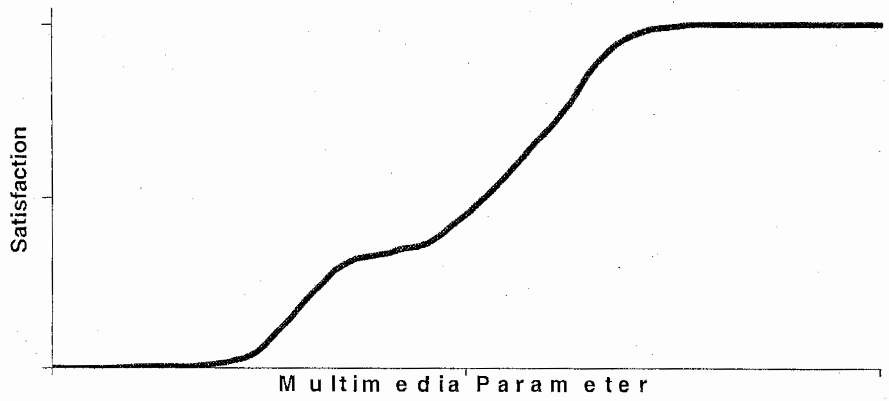

Fig. 6. The general shape of the satisfaction function with regards with one multimedia parameter.

plies a good balance between the values of different parameters. Indeed, multimedia aspect in a usual hardware and network environment is determined by the dissatisfaction caused by the low quality of some parameters rather than by the good quality of the others.

For example, if the image quality is very poor, no matter how you increase the frame rate, the aspect of the video will remain poor and the satisfaction level low. This leads us to think that to increase satisfaction, it is necessary to increase all the parameters. As available bandwidth varies, the way the user decides each parameter's evolution relatively to the others determines his/her profile. Nevertheless, all users will unconsciously tend to tie the value of each parameter to the others.
As a consequence of the two above paragraphs, the bandwidth is a complex non-linear function of satisfaction. In particular, it increases drastically when satisfaction is high. This is equivalent to say that satisfaction does not vary much when the necessary bandwidth is huge. For that reason, we may want to describe more precisely the evolution of the parameters when there is little bandwidth available and less precisely when a considerable bandwidth is available. This led us to consider the logarithm of the bandwidth instead of the bandwidth itself as an index for the Learning Module. This indeed has several advantages. First as seen in paragraph 4.2, we have defined a constant minimal step to allow quick elimination of incoherent points. Taking the logarithm of the bandwidth as an index implies that for small values, this step represents a small variation of the bandwidth and for big values it represent a bigger one. This provides a mechanism that is at the same time precise for small values and efficient for larger values. Second, the linear approximation being done on the logarithm of the bandwidth, it follows more accurately the shape of the real function (see above). And finally, it allowed a richer representation of the profile as a graph as if using a logarithmic scale, which is less important, but still very useful particularly during the experiments.

4.4 Implementation

The Learning Module is implemented in the PABase class. The inner class PreferedQoS contains a UserMediaQoS object and a SituationDescriptor representing the bandwidth requested when using such a QoS. The points of the profile are instances of PreferedQoS gathered in an OrderedSet. The OrderedSet class is part of the JGL 3.1 package. It is a container in which elements are ordered according to one specified value. In the case of the PA, this value is the bandwidth so that finding the nearest points below and above a bandwidth value is made easily. After having retrieved these two points, the linear approximation between them is performed to find the QoS corresponding to the given bandwidth. The getAdaptedQoS method takes a SituationDescriptor argument and returns the QoS chosen by this method.
The update method is called by the Com Module with a UserMediaQoS and a SituationDescriptor as arguments. This method creates the PreferedQoS containing the two arguments and stores it into the OrderedSet.
The updateIndexes is called when user's best QoS have been changed and therefore all the SituationDescriptors of the OrderedSet must be recalculated. For example if the frame rate is 50%, it does not require the same bandwidth if it is 50% of 15 frames per second or 30 frames per second.
Once again, see the commented sources in appendix for further information.

5 The Communication Module

5.1 Objectives

This module has hardly any processing to do. Its main goal is to transfer the QoS from:

- The GUI to the AA when Set is pushed.
- The AA to the GUI when the system resources change.
- The GUI to the LM when Set is pushed.
- The LM to the AA (and the GUI for monitoring) when the resources change and/or the "Refuse" button has been pushed. Besides this, the only role of the Com Module is to translate QoS when transferring it from the PA to the AA and vice versa.

The Com Module must also provide translation methods from "user level" QoS to "application level" QoS and vice versa.
In order to be reusable with other multimedia data than video, the Com Module must be designed as generically as possible.

5.2 Implementation

The software communication is not part of the researches led in this project. As it is an inevitable problem to deal with as long as you program applications, it has been treated in the simplest way as possible. The Java object-oriented architecture has been helpfully used to achieve this goal, particularly method calls.
To be independent from the type of data processed, we created an abstract class (PACom) from which every specific Com Module should be inherited. In this class, all the headers of the methods of any Com Module are defined. The methods that can be written independently of the type of multimedia data used have been implemented in this class (using the UserMediaQoS and MediaQoS abstract classes). The other methods are declared as abstract methods and have to be written in the data type specific classes.
It is not our aim here to study the source code in detail so we won't describe each method. For further information, see the commented sources or Appendix C. Nevertheless, we can roughly sum up what has been seen in the part 4 and 5 thanks to the Fig. 7. This schema is similar to the Fig. 2 except that for each communication process, the method used are named explicitly.
Each arrow means that the module at its beginning is sending information to the module at its end. The plain arrows represent a call of one of the receiver's methods by the sender. Modules use this to inform others about a change in the environment, whether it be a modification of the network condition or an intervention of the user. The dotted arrows represent a call of one of the sender's methods by the receiver. In this case, the information is the return value of the method. This happens when the PA has to consult its Learning Module after to adapt to a new situation. Note that the SetQoS method of the AA is used in both directions: its argument informs the AA of the new QoS to use and its return value informs the Com Module of the resources needed for such a QoS.
All these communication methods only manipulate abstract QoS classes without having to know what kind of QoS it is. The PACom abstract class implements them all. Only the translations are highly dependent on the type of data and even on the hardware used (e.g. screen definition).

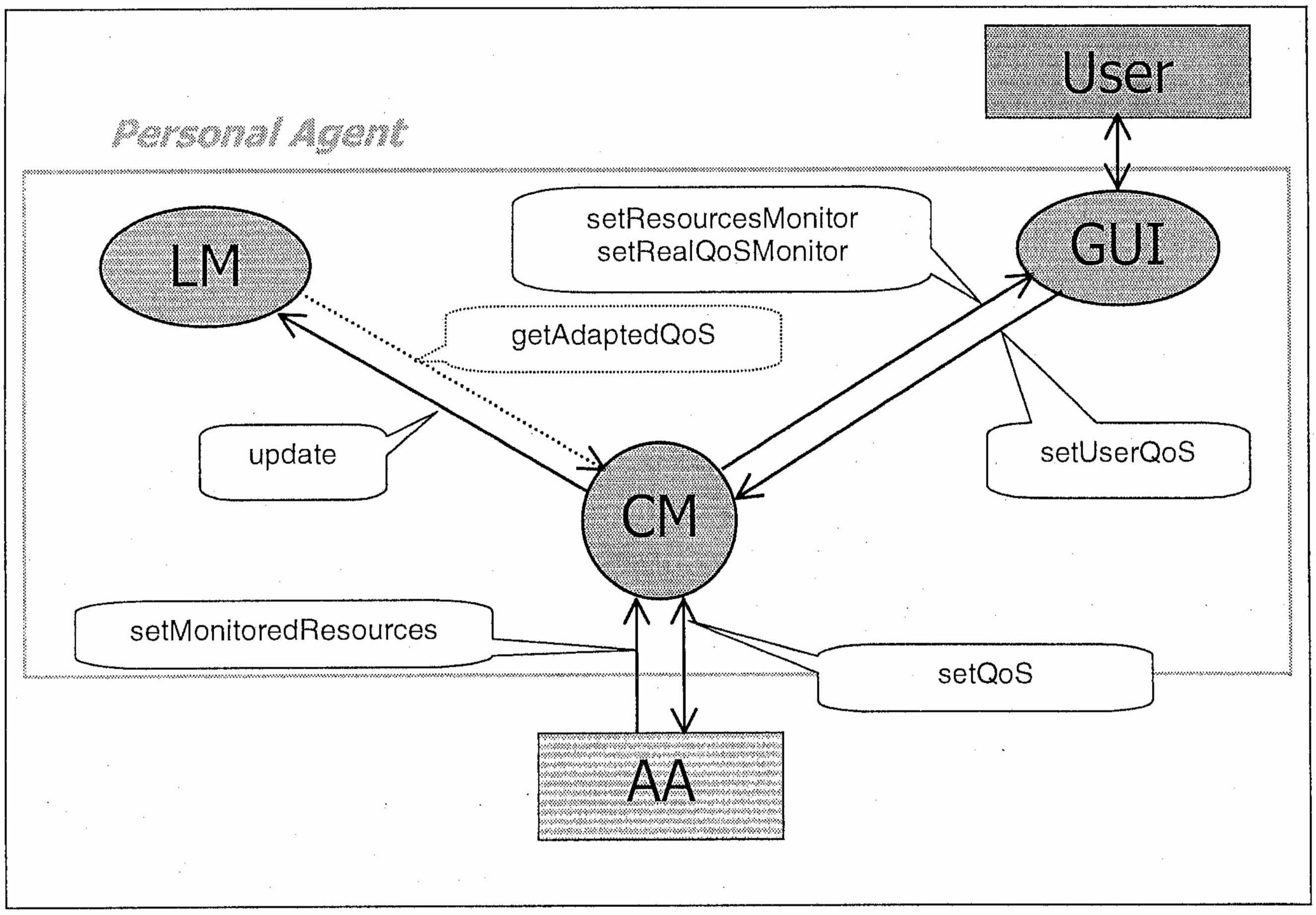

Fig. 7. The Java methods used for internal and external communication.

6 Experiment and results

6.1 Experimental protocol

To test and validate the PA as defined above, two computers linked by an ATM network have been used. Both were equipped with JPEG boards. The first computer, the server used its board to convert a video signal to an M-JPEG stream. An application written in C++ was used to send the video to the second computer, the client, through the network. Due to the high capacity of the ATM network used and the closeness of the two computers, we assume that this system was able to transmit almost any quality of video.

The real AA was still under development when the tests were conducted. Network monitoring functions were not available yet. The limitations in the bandwidth, necessary to test the adaptability of the PA, were therefore simulated. We programmed an AA Emulator, in Java. As it ran on the same Virtual Machine as the PA, the communication could be made through simple method call. The settings given by the PA were really sent to the video application through a Java Native Interface method. A C++ function was added to the application and was called from the AA Emulator Java code. However, instead of being measured

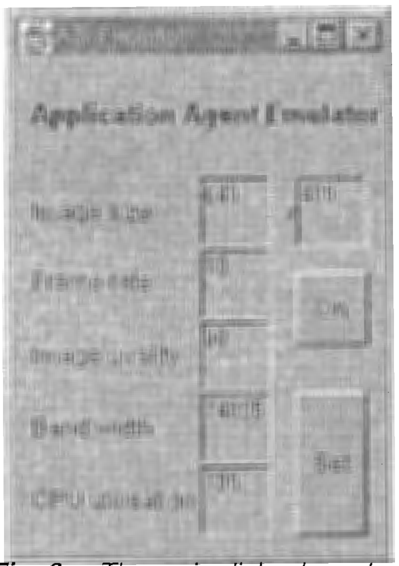

Fig. 8.a. The main dialog box of the AA Emulator, showing the current situation and settings.

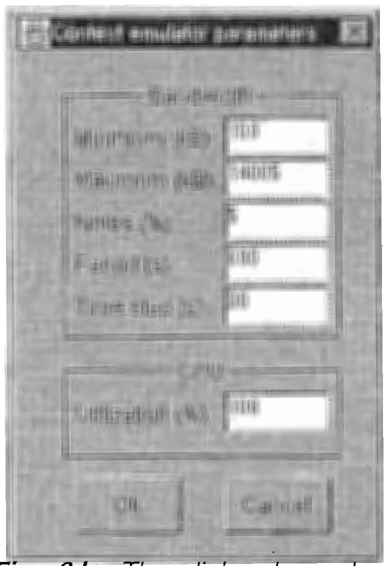

Fig. 8.b. The dialog box allowing the user to set the characteristics of the simulation.

by hardware monitoring agents, network throughput (called bandwidth in the application) and CPU utilization were directly input by the user or calculated by the AA Emulator. As far as CPU utilization is concerned, it appeared that the user could enter the value directly. Indeed CPU utilization changes only with local applications and does not vary much during the viewing of a multimedia presentation. Moreover, we have not used this value to index the QoS in the Learning Module so far, so that it is not taken into account in the tests described in this document.
There are complex model used to simulate the behavior of networks[5]. In our case, such algorithms were not necessary as we only consider the state of the network from user's point of view. Only the throughput, what we called the available bandwidth, had to be modelized. The available bandwidth was implemented to be able to change automatically during one simulation. The function describing the bandwidth has two components: a periodic function and a random parameter. Each of them allowed focusing on different kinds of multimedia applications.

Simulating non-stop running applications and offline learning

The periodic function can be used when one wants to cover quickly a large range of situations, for example you can emulate the behavior of an application running 24 hours a day on the Internet. The traffic on the Internet has a period of 24 hours and varies according to telephone rates, working hours, etc. Generally it follows a pseudo-sinusoidal curve. In our simulations, due to the non-linear relation between the bandwidth and the aspect of the video, we decided to insist on low bandwidth and to consider few points in high bandwidth. In order to do so, we used a function defined as the absolute value of a sine. Taking the opposite value of such a function leads to have sharp peaks for high values and sinusoidal shape for low values (see Fig. 9.). Such a behavior allows rather quick learning (as we will see in the results) but does not correspond to the real conditions under which multimedia are being used. For example for a video on demand service, it is reasonable to consider

that users will watch the videos at approximately the same time or so and that they won't watch them all day and night long.
Nevertheless, such simulations are not meaningless. They can be understood as an offline learning process. We could implement the PA so that when using it for the first time, the user would spend 10 minutes or so looking at a sample video and making the PA learn his/her preferences. This is not the objective as we defined it first, where the PA had to use online Learning Mechanism but still, these experiments show the results of such a use of the PA as we programmed it.

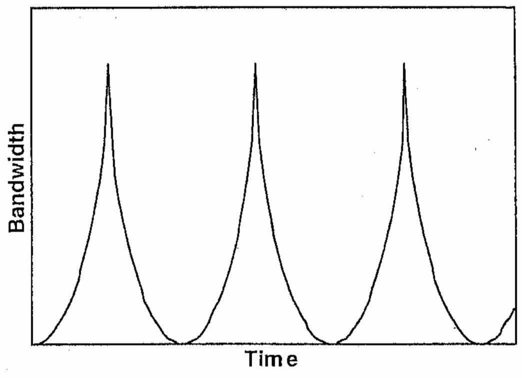

Fig 9. The evolution of the bandwidth with time during the simulations without noise.

Simulating Internet-based multimedia applications and online learning

Nevertheless, we also want to test the PA in conditions that reflect the real utilization of multimedia applications using the Internet. Multimedia applications mainly based on video aimed at a large audience include video-on-demand, teleconference and video-chat services, etc. Usually, such applications are running on short periods of time, compared with 24 hours. Consequently, we can imagine that during one "session" (period of utilization), the bandwidth will not change much. But the fact that from one session to another, the bandwidth can be completely different justifies the need for adaptability. In that case, simulation should present randomly chosen situations without following a periodical function. Each time a new bandwidth is given, a different session, at a different time in different conditions, is simulated. In order to make simulation short enough (8 minutes), each session will last about 30 seconds. Contrarily to non-stop running simulations, the whole range of bandwidths may not be covered in one simulation. This depends to the randomly chosen values of the bandwidth. Learning may therefore take more time. Of course the two approaches can be combined to generate a periodical function perturbed by noise.

Measures

In order to utilize the experiments, two kinds of data are at our disposal. First, we can utilize user's profile itself, which can be compared to others, whose evolution can be observed by saving it after each simulation, etc. To evaluate the adaptability of the PA, we also memorize the time (from the beginning of the experiment) each time the user presses the Set button. This shows us if and when the user has to intervene by him/herself.

6.2 Experiment-based adjustment of the PA

When beginning to use the PA for demonstrations and having it used by other people, several changes had to be made. To take into account the practical differences between application settings and human perception of the video, experiment-based adjustments concerned mostly the quality of the image, corresponding to the JPEG quantization factor. Another important point was the value used by the Learning Module as an index to store, retrieve and interpolate the QoS, according to user's preferences.

The sets of values

In order to make the settings more understandable to any user, the number of authorized values for each parameter had to be carefully chosen. A too big number of possible values for any parameter would lead the user to wonder what effects a change of this value has on the video. For example, passing from 9 to 10 images per second does not provoke a notable change in the aspect of the video. A too small number would not provide sufficient precision and would constraint too much the profile so that all users would have a similar profile. It appeared that a 4-value set is the best compromise, making the difference between two steps clearly understandable without losing too much precision.

The quantization factor issue

As far as the image quality is concerned, not only the number of values had to be determined but also the correspondence between this value and the JPEG quantization factor had to be modified. Although other parameters can be proportionally calculated with regards to a reference, the relation between the quantization factor and the aspect of the image is quite non-linear. Indeed, when the quantization factor is small, small changes in its value imply quantitative differences in the aspect of the image. At the same time, the necessary bandwidth hardly changes. On the contrary, when the quantization factor is between 30 and 100 (its maximum value), it hardly affects the aspect but scarcely influences the size of the data and therefore the necessary bandwidth. Because of these facts, the values of the quantization factor corresponding to the four image quality values have been determined in advanced, through experiments. They represent four really different qualities, in other words, four different visual aspects, of an image.

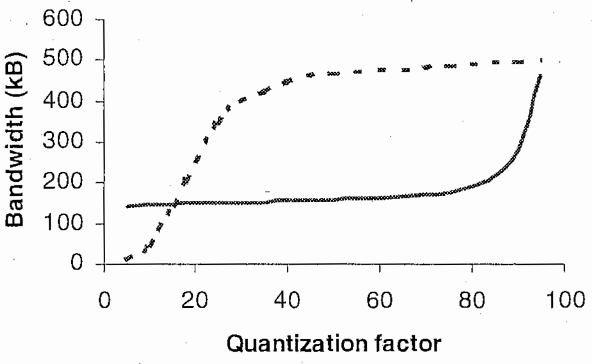

Fig. 10. The requested bandwidth with respect to the quantization factor for a 160x120 and 15-image-per-second video. The dashed curve gives the shape of the function representing the subjective quality of the image (in arbitrary units).

How rounding affects adaptability

Reducing the number of values induces a quantitative gap between two contiguous values of a parameter. As a consequence, the indexes corresponding to two sets of QoS parameters are considerably different. So when linearly interpolating between two points the way calculated values are rounded is crucial. This is even more important as the "logarithmic index" (see 4.3) provides an interpolation that is particularly near the real function. If the usual approximation of taking the nearest integer is used, when descending along the indexes from a point set by the user, the following behavior is observed. First, the index of the approximated QoS is below the available resources' index. Then, as the situation worsens, as long as nearer from the higher integer than from the lower, the QoS remains the same so that the available resources' index gets lower than this QoS index. When approaching the lower integer, the parameters decrease and the index becomes lower than the current situation. We must avoid the phase in which the index of the QoS is higher than

the really available resources. We could use a "floor" approximation, that is to say always round to the lower integer and so the lower point. Unfortunately, this provides a too "pessimistic" behavior. As soon as a point set by the user is passed (when descending the indexes), the QoS parameters decrease even if the previous QoS could still be provided by the network.
The solution is simply that when the Learning Module interpolates the QoS parameters, it first uses a ceiling approximation. If, and only if, the index corresponding to this QoS, calculated by the AA, is above the current resource level, it computes an interpolation based on a floor approximation. In that way, the change from one QoS to the other is done exactly when the available resources allow it. Consequently, the PA follows more accurately the situation of the system and less intervention from the user is required.

6.3 Results

At the time this document has been written, all the experiments could not be completed. That is why complete results cannot be provided. However, from the first experiments, the following statements can be made.

Offline learning simulation

When using a periodical function for the available bandwidth with little or no noise, rather quick learning was achieved. The period of the function was 10 minutes and the value of the bandwidth was changed every 30 seconds. In half a period, the whole range of values has been covered and the profile has almost its definitive shape. In a whole period, a few final adjustments were made. From the beginning of the second period, the users no longer needed to set the values. Furthermore, the profile constructed provided the requested behavior in any random situation.

Online learning simulation

These results are not available yet.

6.4 Future research directions: experiments and improvements of the PA

A good source of material: proposed experiments

This first version of the PA has several drawbacks that must be repaired before actually using it. Nevertheless, it allows us to conduct experiments in order to create a large amount of user profiles. This has to be done with many different users, using various kinds of video, in different situations, etc. Significant information should be extracted from the so built database. For example are there really considerable differences between users. This is the key point to choose between the two approaches quoted in paragraph 2.1. If all the profiles are alike, it is plausible that a universal function can be found to describe user's satisfaction to a set of multimedia parameters. In that case, the data obtained through the PA could be used to approximate this function. On the contrary, if there are big differences between the profiles, the second approach should be better. Once again, data collected through the PA could help to refine profile definition, eventually by showing categories of users or fundamental characteristics of profiles. Another point that has to be explored is the influence of the content of the video in user's preferred settings. On that particular topic, the researches of Apteker et al.[6] led to a classification of videos in eight categories, according to temporality, audio and visual message content. Through their own experiments, they showed the dependence of video's watchability (in other words user's satisfaction) on the content of the video. Further researches should explore the balance between the different parameters rather than the influence of one single parameter (in [6], the frame rate) on watchability. Experiments using the PA, by showing the shape of user's profile in each category would shoulder these researches.

The GUI is not user-friendly enough

Using the PA even for people familiar to computers has proven difficult. This comes from the fact that we designed the GUI with regards to the theoretical profiling and storing process. As a result and paradoxically, the user has to learn to adapt him/herself to the PA although the goal of the PA was precisely to adapt to the user. Of course technical aspects, as for example JPEG's quantization factor, have been eliminated and easily understandable notions are used, but still the interface is not user-friendly enough.
Generally, the user has to deal with a Set process that interferes with the manipulation of the main application.
For example, he/she has to ensure "manually" that the resources are completely used by looking at a progress bar. During the experiments, the users were tempted to choose the QoS not with regards to their preferences but in order to use the available resources as much as possible. This can also be explained by the fact that they had no particular interest in the video they were looking at. Anyway, some users spent more time looking at the resources progress bar than watching the video. This point leads us to think of a way to allow the user to manipulate the QoS parameters at a given resource level. Instead of manipulating the somehow abstract notion of resources utilization, the user should only give the importance of each parameter relatively to the others and then the PA would set the parameters by itself in order to use all the available resources. Just as if when moving one slider, the others were automatically moved by the PA in order to keep the index of the QoS constant.
Concerning the human-PA interaction, we must take another point of view. It is necessary to think of what kind of interface the user would like to use instead of focusing on the learning process. We have to consider that the user refers to a particular situation when changing the value of a parameter. In the PA as it is implemented now, the references are too complex and abstract: the best QoS values, the progress bars showing the current settings, etc. The simplest reference for the user is not any monitored value but the multimedia event itself,, i.e. the video, the sound, etc. It is far easier for the user to say what has to be improved in the current situation than to set values even with sliders. For example the user could indicate which parameter is not satisfying and has to be increased (or decreased). It is more natural for a person to say "I would prefer this video to be smoother" than "We must increase the frame rate by 10 percents and decrease the quality by 5 percents so that the network can still afford it". Taking into account user's point of view more than programmer's is the key point to improve the GUI of the PA.

The PA is too dependent on the hardware

In the future, the PA could be implemented on a Portable Digital Assistant (PDA). The user would be able to connect it to any computer or other machine before utilization. Through this connection, the terminal would learn user's profile and provide an interface and multi-media events that fit user's tastes. This new idea has revealed another weakness of the PA. The data, even if it is expressed in percentages or other units distinct from application's settings, is completely linked to the hardware. For example, the fact that the bandwidth (even combined with other parameters as CPU utilization) is used as an index to choose the correct QoS prevents us from using the same database on another machine that would have other characteristics. In fact, our PA is mixing the user aspect and the application aspect. This partly comes from the fact that it is in charge of the translation from user QoS to application QoS. Because of this translation, a reference QoS system had to be implemented and the PA has to deal with application QoS. This assumes that the PA is aware of the characteristics of the hardware and the software being used. This "case by case" translation prevents a proper separation of the function of each agent and a really hardware-independent PA. Rather than that, the PA should send a complete user profile to the AA and the AA should be in charge of exploiting this profile according to the available resources. In that case, the PA would never be aware of the current available resources nor the multi-media characteristics of the hardware and software system. For further details on a hardware-independent user-profile, please read the note in appendix.

7 Conclusion

One month before the actual end of this internship, we can only draw temporary conclusions. Due to the recentness of the project, the definitive design of the whole application has not been set yet. The demanded task was to investigate the possibilities of adaptation to the user in a previously defined context of agents. An implementation was also required to be able to test the theoretical ideas and eventually the whole QoS management system as soon as possible. Unfortunately, it has not been possible in three months to gather the different agents of the system.
The work done on the PA allowed us to find general ideas and research directions for the future more than to develop a definitive PA. This research project can be seen as a first approach of the problem. Appendix A gives an example of studies that can continue this work, taking into account the remarks made in paragraph 6.

Appendix A. References

[1] K. Gerwig. “Business: the 8th Layer”. *Networker*, Ferbruary/March 1998

[2] A. Vogel, B. Kerhervé, G. von Bochmann and J.Gecsei. “Distributed Multimedia and QoS: A Survey”. *IEEE Multimedia*, 2(2), Summer 1995.

[3] G. Weiβ. “Adaptation and Learning in Multi-Agent Systems: Some Remarks and a Bibliography”. *Adaptation and Learning in Multi-Agent Systems, IJCAI’95 Workshop*, August 1995.

[4] R.T. Apteker, J.A. Fisher, V.S. Kisimov, and H. Neishlos. “Video Acceptability and Frame Rate”. *IEEE Multimedia*, 2(3), Fall 1995.

[5] Cooperative Association For Internet Data Analysis. Web site: http://www.caida.org.

[6] A. Richards, G. Rogers, M. Antoniades, and V. Witana. “Mapping User Level QoS from a Single Parameter”. *Second IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, Versailles,1998.

Appendix B. A HARDWARE INDEPENDENT USER PROFILE FOR THE PERSONAL AGENT

ACR LABS. INTERNAL PAPER.
AUTHOR: ANTOINE RAUX

INTRODUCTION

A complete Quality of Service management system should not only be an improved monitoring application requiring the user to set the parameters of the applications according to the QoS. It should also learn about the user to be able to set these parameters by itself, according to the user's tastes. A multi-agent structure is particularly adapted to this context. The Personal Agent would be the "companion" of the user, and the other agents (Application Agent, Stream Agents, etc.) would be in charge of the "non-human" part of the problem: software and hardware. In our system, the Personal Agent is designed to be *mobile* (for example implemented on a PDA). That means that the user, carrying his Personal Agent with him, could get the multi-media applications to fit his preferences when using any type of computer (or any kind of electronic machines, games, handy-phones, cash dispensers, etc.).
The same application settings do not give the same feeling to the user when running on different systems. For example, an image of 256 per 256 pixels has a different size in centimeters when running on a high-resolution screen or a low quality LCD display. The system's characteristics such as bandwidth or CPU are other hardware dependent parameters. A high-class desktop computer and a portable visiophone will not be able to display a video with the same quality in the same conditions.
This implies that the representation of user's profile in the PA must be completely independent of the system on which the multimedia application is running. In other words this representation must be *abstract*, its meaning must be closer from the human way of thinking than from the computer's one. The PA must deal whether with "human physical units" such as centimeters or with completely general terms that still have to be universally defined; for example "big", "small" or "smooth".
The PA would determine all these abstract values thanks to a learning process and send them to the AA. The AA, being closely linked with the hardware characteristics of the system would be in charge of getting the QoS corresponding to one *situation* and translating it from "human values" into "computer values" (e.g. centimeters into pixels).

THE QoS SELECTION PROBLEM

At one particular moment, a QoS administration service must deal with a QoS level that is imposed by the exterior (the ISP, according to the kind of contract you have for example). To this QoS level corresponds an amount of available resources that we call the *credit* of QoS. We must choose a QoS whose *cost* (in terms of resources) equals that *credit*. Once you have the credit, there are still a lot of different QoS to choose from; for example, some of them will have a big size and a poor image quality, others a good frame rate but a small size, etc. The goal of the PA is to choose among these different configurations the one that the user will *like* the most. We found that a good representation of the different costs of the QoS would considerably help to do so.

The representation we have chosen is an n-dimensional space, n being the

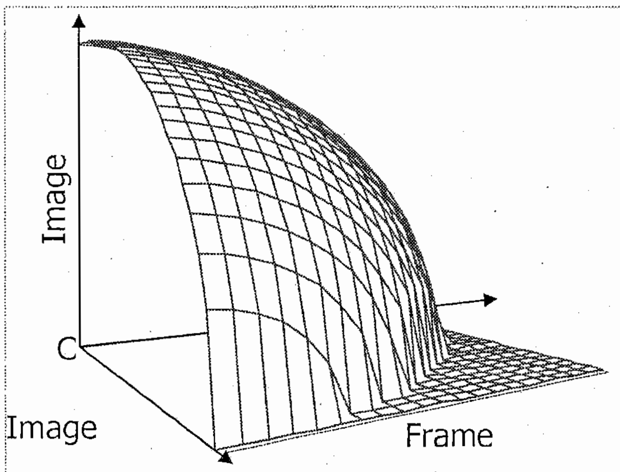

Fig. 1. *An isocost in the case of a 3-parameter QoS (here a video sequence, in arbitrary units).*

number of parameters of the media considered. For example, in the case of the video we can at first consider three parameters: image size, frame rate, and image quality. A fixed cost gives a constraint and so reduces the number of dimensions. We define *isocost* as the hyper-surface (of dimension n-1) made of the points that have the same cost. In our example, with 3 parameters, it would simply be a usual surface inside the usual 3D space. The problem is then to choose one point of this hyper-surface.

MATHEMATICAL BASIS

A good way to parameterize an isocost is to change the coordinates system to hyper-polar coordinates (in 2D, polar coordinates; in 3D, spherical coordinates). This is mathematically possible as long as you assume that in any direction, the cost is strictly crescent with respect to the distance to the origin. This hypothesis is reasonable because the cost increases when you increase any of the parameters without changing the others. For example, if there are two parameters represented in a planar graph, let's call P the point representing a QoS and O the origin of the coordinates. We characterize the position of P through the angle θ between (OP) and one of the axis. In the case of a three-parameter video sequence, we do not give the position of a point P through its values in terms of image size, frame rate, and image quality but thanks to two angles, θ and φ. θ is defined on the projection of the surface on the frame rate/image quality plane. Let's call P' the projection of P in this plane and let's call O the origin of the space. θ is the angle made by the (OP') line and the frame rate axis. φ is the one between (OP) and the image size axis. In the general case, there must be n-1 angles to define the position of the point (if n is the dimension of the space i.e. the number of multi-media parameters).

PRACTICAL MEANING AND NECESSARY ADAPTATIONS

Practically, these angles allow us to compare the parameters with each other. For example, when θ, as defined above, is small, the point is nearer from the frame rate axis than from the image quality axis. Therefore the frame rate will be "bigger" than the quality (if ever we could compare these two values). When φ is small, P is near from the image size axis so the frame rate and the image quality are "small" compared with the image size. The advantage of this technique is precisely that it allows us to compare such different notions.
To maximize the stored information, a good choice of units is necessary, including non-linear transformation. For example, the quantization factor characterizing the image quality of a JPEG image is completely non-linear with the impression given and the cost. When very small, the aspect changes much and the cost remains very low. On the contrary, when approaching 100, the aspect hardly changes although the cost increases drastically. If the PA is to deal with values characterizing the *aspect* of the image, the AA will have to be able to translate an "aspect value" into a quantization factor in a non-linear way.
Although angles are a nice mathematical approach of the problem, their use implies trigonometry which computers do not feel comfortable with. The following algorithms will consider slopes. Slopes are the analytical equivalent for geometrical angles. The angles are better in graphs whereas slopes are a good way to compare values. For example, instead of using the θ angle, we will use the ratio between the frame rate and the image quality. We can also replace φ by the ratio between the image size and the image quality for example, which is not completely equivalent but gives as much information. In a nutshell, we need n-1 ratios to characterize a point located on a given isocost of the n dimensional QoS space.
These ratios, *known by the PA*, describe a straight line in our n-dimension space. One isocost (which is hardware dependent and though *only known by the AA*) corresponds to the available credit. The chosen point will be the intersection between the line, which represents user's priorities and the isocost. We have therefore theoretically solved the problem of finding the unique point maximizing user's satisfaction at a given cost. Unfortunately, theory and practice are two different things and before implementing or simulating anything, some obvious issues must be treated.

THE TWO TYPES OF PARAMETERS

The main problem resides in the paradox that too much adaptation finally dissatisfies the user. For example, in a highly variable network, the user will not be satisfied if the size of the image changes every second, even if each size in itself

corresponds to what he would like to see. In this case, it is perhaps more comfortable to have a smaller but constant size. On the contrary, the frame rate should be as high as possible, according to user's priorities, however variable it is. These simple statements lead us to consider two types of parameters:
Pseudo-Continuous Parameters (PCP), which must be maximized as much as possible.
Discrete Parameters (DP), which can take only a few values and must not change too often.

A SIMPLE REPRESENTATION OF USER'S PROFILE

The DPs are characterized by *thresholds* rather than by ratios. A threshold indicates when a DP has to be decreased by one step. The problem is what kind of values do these thresholds have? As we said before, no such information as the cost of a QoS or the surface representing this cost are stored in the PA. Therefore we based the thresholds on the PCPs' values. Indeed, the user does have minimum requirements as far as the PCPs are concerned (few people would like to watch a film with a frame rate of 1 image per second, however big the image is). As long as the AA can adapt the PCPs to fit the available credit, the DPs remain the same but to prevent the PCPs to become to small, a limit has to be set. Whenever the PCPs reach this limit, the DPs have to be decreased. Therefore the thresholds are the link between the PCPs and the DPs.

USING USER'S PROFILE

To make it brief, the values necessary to describe a profile contain at least:
- For the PCPs (we assume there are p of them):

p-1 ratios to describe their relative importance
a minimum value (*threshold*) under which they must not be
- For the DPs:

a way to know when each value must be increased or decreased by one step.
We can simply determine this by ordering the DPs according to their importance.

At the beginning of the connection between the PA (on a PDA for example) and the AA (on any computer-like machine) the PA sends these characteristics to the AA. During the utilization of the application, the behavior of the AA would then be the following:
At a given set of (n-p) DPs, in the p-dimensional space of the PCPs,

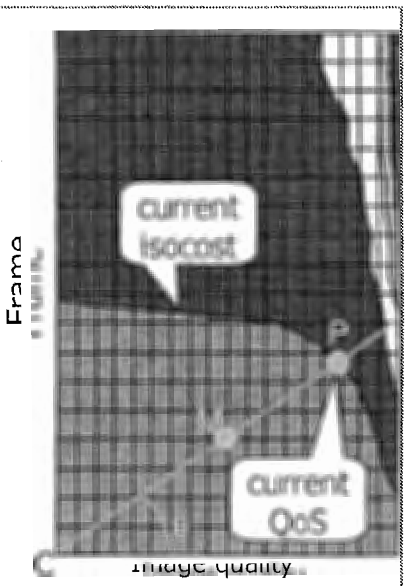

Fig. 2. *The PCP graph in the case of a two-PCP QoS. (here for a video sequence, using arbitrary units).*

let's call (D) the straight line containing the origin and described by the p-1 ratios seen above. Let's call M the point of (D) that represent the minimum values of the PCPs and P the point describing the current QoS. At first, P is *above* M (that means further than M from the origin). As the cost decreases, P descends on (D) towards M. When P reaches M, a new set of DPs has to be chosen. Here, we choose to decrease the *less important* DP (as determined by the order of importance seen above) by one step. Therefore, to fit the given cost, the AA increases the PCPs. P ascends on (D). As the cost decreases, P descends on (D) until it reaches M again. Then, for example, the AA decreases the second less important parameter by one step *or* the first less important parameter by one more step (according to the algorithm chosen). And so on.

THE LEARNING PROCESS: MORE COMPLEX MODELS MAY BE NEEDED

The simple model is quite easy to implement but it may not always satisfy the user, particularly concerning the choice of the DP to decrease. A single ordered list of the DPs might not reflect user's tastes. The learning process itself incites us to change the model. Indeed the simple model would be practical if the user indicated the ratios, thresholds, DPs' order, etc. through a form. But as we have stated in introduction, the goal of the PA is that the user does as few conscious teaching actions as possible. To fulfill such an objective, it is natural to use *learning by example*. The user should just tell what he wants to see while using normally his applications and the PA must modify its parameters according to it so that fewer and fewer manual settings are needed. A most user-friendly interface would simply allow the user to say which parameter does not satisfy him and has to be increased (assuming that a higher value is always more satisfactory for the user). Consequently, the PA decreases the value of the other parameters by itself, in order to keep the same cost. In such a context, the DPs might not just be ordered but the PA could have to store when each DP has to be decreased case by case. The thresholds and even the ratios could be variable. Nevertheless, this would imply more complexity as far as implementation is concerned but the principles would be the same.