QoS Control: an Application Integrated Framework

João Bom[2], Paulo Marques[2], Miguel Correia[3], Paulo Pinto[1,2]

[1]Instituto Superior Técnico, Lisboa
[2]INESC, R. Alves Redol, 9 P-1000 Lisboa, Portugal
[3]Universidade de Lisboa, Faculdade de Ciências, Lisboa
{joao.bom, paulo.marques, paulo.pinto}@inesc.pt mpc@di.fc.ul.pt

Abstract: Network technology is becoming simpler in order to support the widest possible range of user applications. Therefore, the specific requirements of each application must be performed outside of the network and the adaptation mechanisms should be done by the applications. For the case of multimedia applications the programming task can even be more simplified if several “low-level” aspects can be made transparent by the use of a suitable framework. This paper presents such a framework with some algorithms to control some of the previous aspects, namely data handling and dynamic QoS control. It provides the application with a mechanism to constantly adapt the quality of the continuous media being transmitted. The algorithm takes into account not only the network conditions but the end host conditions as well. Some experiments are presented.

Keywords: Distributed Multimedia; Quality of Service; Dynamic QoS Control; Adaptation Algorithms

1. Introduction

On the road to construct future-proof multiservice networks, the solution has been to reduce as much as possible the features provided by the network itself. Frame relay networks started this approach and ATM networks followed suit. Therefore, all the application requirements have to be handled at the end machines, outside of the network. The variety of application requirements is large and the new Transport protocols have already a structure where different behaviours can fit. The interface to the network is based on contracts about specific semantics of the traffic (constant or variable bit rate), its real-time characteristics and the amount of bandwidth needed. This type of interface is still at a too low-level for multimedia applications and the concept of Quality of Service (QoS) is seen as appropriate to fill in the gap.

On the other hand, multimedia applications have a strong component on continuous data handling (audio and video), some of which is tedious and complex for a direct treatment by the application but could easily be delegated to autonomous components under application supervision. Multimedia data has its own high level semantics (think about M-JPEG or MPEG) and the adjustments to the working conditions can be characterized easily between the application needs and the component behaviour. This division has another advantage: the logical part of the application is programmed as if the resources were unlimited and the QoS parameters will adjust to the reality. This enables the use of the same application over network and machine technological improvements and eases the construction of new application because only the logical part and the interface to the components have to be done. This division also fits well with the characteristics of most of the current operating systems. They provide a best effort service. Their processing speed depends on the number of processes and on their "weight". A heavy load on the host can seriously affect its speed in treating multimedia data. Under these assumptions, it is natural to think on using the networks as if they are also offering a best effort service, always trying to get the most out of the current conditions. ATM networks are fairly appropriate to multimedia data because they support variable bit rate and can allow the user to overpass the contract at his responsibility. Due to the characteristics just outlined this responsibility aspect is not a problem. Moreover, traffic control on ATM networks is very conservative [1] leaving enough resources to use at application discretion.

This paper proposes an integrated framework to develop multimedia applications which takes care of the tedious parts of data handling and controls the QoS. There is a division between generic issues and issues that depend on the specific media being managed, that have to be considered whenever a new media is introduced. The data handling part includes the capture, the sending and receiving to and from the network, and the presentation of a media without a direct intervention of the application.

The QoS aspect has a feature we consider important. QoS is expressed in terms of QoS parameters. We allow the programmer to express the QoS (s)he wants in terms of QoS parameters that are meaningful to him and the application (for example: frame rate, window size, latency); the architecture translates these parameters into system parameters like packet loss and jitter.

The specific QoS adaptation algorithm proposed here defines a scale of QoS working levels and the framework component increases or decreases its level according to the network and the end systems conditions. The application can be notified about the work being done on its behalf.

For instance, if the lowest level of the scale is reached and the system should have even gone lower, a decision, like terminating the connection, has to be made by the application.
The inclusion of the end hosts conditions in the adaptation algorithm, another feature not seen on other works, makes sense because the machines are one of the bottlenecks when using new networks, such as ATM (as the experiments proved).
The paper starts with some brief considerations about Transport levels and ATM network usage. Then, a brief description of the framework is given followed by a description of the adaptation algorithm. The rest of the paper presents some results of the experiments, analyses the related work and draws some conclusions.

2. Protocols and networks

TCP is not suitable for continuous media transmission for some reasons. Firstly it imposes its own transmission rate to the data, not the one that the application wants and needs. The "slow start" and the congestion control mechanisms are the main reasons for this behaviour. Secondly, TCP delivers all data without errors, doing retransmissions if necessary. This is not needed for multimedia data and, in general, retransmitted data will not be delivered in time for presentation, consuming extra bandwidth and processing power, possibly increasing congestion and losses.
New protocols, more adequate for continuous media data, started to appear. The Real Time Protocol, RTP [16], is one of these protocols and is the one we used in the experiments. More than the specific choice of the RTP, the following four ideas make it adequate for multimedia data and should be present in Transport protocols that can handle application requirements outside of the networks:

1. RTP, does not impose an error correction mechanism. It has a control protocol, RTCP, that exchanges monitoring information like the packet loss rate and the interarrival jitter between the sender and the receiver ("receiver reports" and "sender reports"). Although the standard does not propose any control algorithm, the adaptation algorithm of our framework can fit in this place.
2. It uses Application Level Framing (ALF) [7]: the unit of communication has a meaning to the application. For example, for video there is the knowledge of a frame and not only a couple of segments.
3. It uses Integrated Layer Processing (ILP) [7]: the processing overhead caused by the existence of several layers is not acceptable for multimedia so an integrated layer is used. Normally RTP is implemented as part of the application. We do not use this approach because we do not want the application to manage multimedia data directly.
4. RTP is a generic protocol, independent of the media being transported. The aspects related to specific media types are defined separately in profiles [15].

A natural way to use ATM networks as a best-effort service might be the Available Bit Rate (ABR). However, ABR was designed to serve data applications that can control changes on bandwidth usage. It has a strict control mechanism based on cells (not video frames, or audio spurts) and is very strict about losses (data applications are rather sensitive to cell losses). Clearly, ABR is not suitable for multimedia. A better approach is to use Variable Bit Rate (VBR) services and work over the contracts using priorities (CLP=1). We assume that the UPC/NPC mechanism uses cell tagging until the Peak Cell Rate (PCR) value and discards cells after that. Therefore, a video connection, for instance, should set the PCR in accordance to the frame length, but can set the Sustainable Cell Rate (SCR) to a value lower than the actual value it intends to use (just above the minimum QoS level, for instance).

3. Management framework

The management framework’s main goal is to simplify the programming of distributed multimedia applications.
More specific goals are: (1) to manage the QoS in a best-effort environment; (2) to handle the multimedia data on behalf of the application (capture / send / receive / present); (3) to hide the lower level aspects permitting the application to work only with higher level QoS parameters.
Although the framework handles autonomously most of the operations related to continuous media, the application can have different degrees of supervision over this management by exchanging information with other components. Figure 1 shows the various components of the framework and a brief description of the more important ones is given below. A more complete description of the framework can be found in [2].

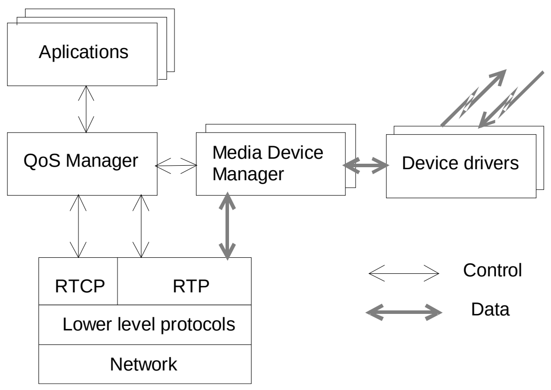

Figure 1. Management framework: configuration of the components in one of the system’s hosts.

The *QoS Manager* and the *Media Device Managers* are the main entities to accomplish the framework goals. The Device Drivers are just wrappers around the video (image, audio, etc.) input and output device drivers.

3.1 Media Device Managers

The Media Device Managers (MDMs) manage a media on behalf of the application. At one host the MDM gets media samples from an input device driver and sends them to the network (to RTP); at the destination host it gets the data from the network and delivers it to the output device driver. These components impose the schedule of capture and presentation; they do some buffering to filter jitter; and they do all the other tasks that the application would have to do to deal with the media samples.
The MDMs are media specific. There is one MDM per type of media: MJPEG video, MPEG video, ulaw audio, etc..
MDMs do not control QoS. This is performed by the QoS Manager that just informs the MDMs of the decisions it took. It is responsibility of the MDMs to execute these decisions in a medium specific way. At the beginning of a connection it is also the MDM that informs the QoS Manager of the values to be negotiated with the network for a certain configuration of QoS. So, the QoS Manager works with general *Programming Level QoS parameters* (frame rate, frame loss rate, Q factor, etc.) whereas the MDMs translate these parameters into *System Level QoS parameters* (packet size, bandwidth, jitter, etc.).

3.2 QoS Manager

The QoS Manager is the entity that does the QoS adaptation to the system conditions in a media independent way. It is where the QoS control algorithm (section 4) is implemented. Media specific valuations and operations are left to the MDMs.
Initially the QoS Manager receives a *QoS scale* from the application with the working levels of the algorithm, and the indication of the initial level. An example of such a scale, expressed simply in terms of the video QoS parameter "frame rate" is: [25, 20, 15, 10] (frames/second). These are the frame rate working levels for the system. The desired value could be 25 fps but 10 fps is still acceptable (each level of the scale can have several parameters, quality factor for video, image size, colour or black and white, etc.).
If the application deals with more than one continuous media it can also define relations and priorities between media streams and sets of media streams, called *composed media streams*. For example, if an application is sending one audio and one video streams it can define that the audio has higher priority than the video so that video QoS will be decreased first, in case the system conditions impose a QoS adaptation.
The application can define the degree of knowledge and intervention in the control algorithm it wants to have. It can tell the QoS Manager which events it wants to be informed of. At least, the QoS Manager always informs about the impossibility to sustain even the lowest level of QoS of the scale requiring an action by the application. The application can decide to provide a different scale with lower levels of QoS, to close the stream, etc.

4. QoS Monitoring and Control

QoS management consists of *monitoring* – determining the QoS the overall system (network and hosts) can deliver – and *control* – adjusting the QoS of the application to that value. The reason to do this adaptation is that asking more QoS than what the system can give, produces a session with lower QoS than the correct one. We prove this statement in one of the experiments of section 5. A brief outline of the algorithm is given here. For a more complete description, see [3].
The QoS monitoring is based on the RTCP receiver reports. The sender sends data to the receivers using RTP. The receiver sends these reports back to the sender. We did not make use of the sender reports of the RTCP.
The algorithm uses three values carried in the RTCP receiver reports. The first one is defined in the standard [16] and is used to monitor the network conditions:

1. **packets lost**: the number of packets lost in the network. The field in the receiver reports carries the cumulative number of packets lost; the packets lost in an interval of time is obtained by subtraction of one value and its predecessor.

The other two were introduced as *profile-specific extensions* in order to measure the conditions at the end hosts:

2. **too_late**: number of packets/frames not presented because they were delivered too late, due to network or end hosts conditions.
3. **nshown**: number of packets/frames not presented due to a heavy load at the receiver that prevented the MDM to get active in time for presentation.

The percentage of the total packets/frames loss in a time interval is given by:

$$\text{losses} = \frac{\text{packets lost + too\_late + nshown}}{\text{total number of packets}}$$

To avoid the control algorithm from being too sensitive to changes, it acts based on a *filtered losses* value:

```
filtered losses = average(last N losses)
```

The control algorithm is a closed control loop. Monitoring gives the value of the filtered losses which is then used to decide the system working level. This working level will influence the number of losses that will be reported by the receiver (figure 2).

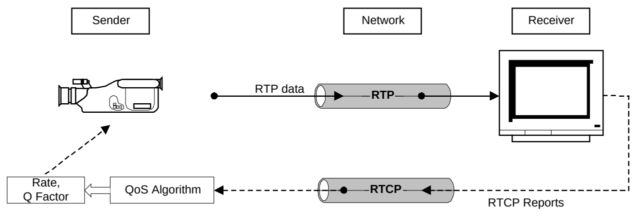

Figure 2. QoS control scheme

As any closed control loop, the round trip delay is important to the reaction time of the system. This aspect will be covered by an experiment.

The control algorithm works in three stages. The **first stage** is used as a first approach to the problem, assuming a transient small problem occurred. This is the normal working stage except when the system is working at its highest QoS level. The action to be taken is to drop a small amount of data. The concrete action is media dependent and is executed by the MDM. An example of such action is the discard of an MPEG B frame or a complete M-JPEG frame.

The **second stage** is entered when the filtered losses passes a higher level λ_s. In this case the working level is reduced to the immediately lower level of the scale, reducing the amount of used bandwidth or demanding less power from the end hosts.

The **third stage** is entered when the lowest QoS level was reached and still another reduction is necessary. This means that the system cannot even sustain the lowest level defined. In this situation the application is informed and decides what to do.

Figure 3 divides the filtered loss values in zones. The *Working Zone* is the zone of values considered to be normal and so no QoS action is performed. The *Transient Problem Zone* is when a first stage action is taken. The *Degradation Zone* is when a second stage correction is performed. The *Improvement Zone* corresponds to the opposite action of the Degradation Zone: to improve the QoS by one level, performed in the second stage.

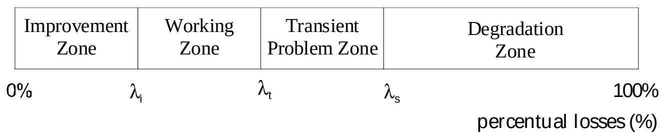

Figure 3. Filtered loss zones.

It is important to refer that the processes of increasing and decreasing the QoS are not symmetrical: a degradation is performed when a problem truly exists; an improvement is performed when it is acceptable to think that there are conditions to do it. It would be difficult to measure if there was really network bandwidth and CPU power to do the improvement because it would need another algorithm and conditions might have changed just afterwards. So, when the losses are "very low" the QoS is increased just to see if a better quality can be obtained. If not an oscillation in the QoS exists but the filtered losses prevent the algorithm from oscillating too much. Anyway, when some oscillations existed, we did not found it to be very disturbing to the user.

The values of λ_i, λ_t and λ_s are obtained experimentally, and are media dependent.

5. Experiments

Our experiments were made using a videoconference prototype on two SUN Sparc 10 workstations with Parallax JPEG video boards connected by an ATM LAN (a single Fore switch).

In the traffic figures of this section the solid lines represent the filtered losses; the point based lines represent the losses; and the trace based lines represent the bandwidth being used by the stream. The two horizontal lines are λ_i (down) and λ_s (up). We used the values 5% and 15% obtained experimentally for these parameters.

The first experiment (5.1) shows two similar sessions with and without the algorithm. The second experiment (5.2) shows the importance of the inclusion of the end host situation in the algorithm. The third (5.3) measures some response times of the algorithm.

5.1 Experiment1: Quality comparison with and without QoS adaptation

The objective of the first experiment was to prove that the system offers a higher quality with QoS adaptation than without it. The experiment consisted of running two similar sessions – one with the algorithm in place and the other one with the algorithm disabled. The first experiment also shows the operation of the system in a typical scenario. Throughout the experiment, the bandwidth was artificially reduced and increased again. It started with no limitations and there was a reduction to 200 Kb/s, then to 100 Kb/s, increasing after that to 200 Kb/s and again to a situation without limitation.

The scale of QoS used by the application in all experiments is given in figure 4. It is expressed in terms of the QoS parameters frame rate (F.R.) and JPEG quality factor (Q).

Level	1	2	3	4	5	6	7	8	9
F.R.	25	22	19	25	22	19	25	22	19
Q	50	50	50	75	75	75	100	100	100

Figure 4. Scale of QoS (level 1 is the highest quality).

Figure 7 shows the results of the experiment with the algorithm running. The initial level was level 5. In most of the experiments it was typical to have some unstable period at the beginning because the pipeline of the filtered losses is not full. But right after (t=15s) the losses started to decrease, they went below λ_i and the QoS Manager improved the QoS. Level 1 was reached at about t=45s and the used bandwidth was high. At t=60s the bandwidth was reduced to 200 Kb/s. The system was using more bandwidth, so losses started to happen and the system moved to level 2 and to level 3. This move was very fast and it moved again to level 2. It stayed in level 2 until t=100s. At this moment, the person in front of the camera moved out to reduce the image complexity. The system went to level 1 and used less bandwidth than before.

At t=120 the bandwidth was again reduced and the losses raised quickly. The QoS Manager moved rapidly to level nine, one level each 5 s (the period used for the RTCP receiver reports) but the system could not even work with this level so the losses never got to a stable value. The third stage of the algorithm would have been performed at this stage. This stage was not implemented so the QoS Manager had to stay in this QoS level despite the losses. During this period the system acted as if the algorithm did not exist.

When the bandwidth was again increased (t=180s) the system took some time to react (the effect of the filtered losses). Losses diminished and when they went below the threshold value the QoS Manager started to improve the quality. When the limitations on the bandwidth ceased to exist the improvement process went on, and the used bandwidth increased as well.

It is interesting to compare the curve of the absolute losses with the curve of filtered losses. They are very similar, although with a certain delay, but the absolute losses curve never increases too much.

Figure 8 shows the operation of the system without the algorithm working. As a first observation the absolute losses curve shows a more erratic behaviour denoting a worst situation for the presentation of the video (the algorithm was just calculating the filtered losses but did not react). The same reductions and increases on the bandwidth were performed. A careful analysis can show that when there were no bandwidth limitations the system did not use as much bandwidth as in the previous case. The reason is that the sender stayed at level 5 (the initial level) not profiting from the better conditions of the system. When the first reduction took place losses increased very much and the sender was not able to decrease its pace. The situation recovered a little bit when the image complexity decreased. The phase with the lowest network bandwidth is similar to the one of the first experiment, and the rest of the experiment is obvious.

A subjective analysis of both situations was made by a user watching the video presentation. The quality in the first case was much better. In the second situation there were many images lost, provoking the freezing of the screen with the consequent disturbing effect. There were two causes for this inferior quality. The first was the presentation of less frames, as it will be shown next. The second is harder to measure quantitatively and is the non uniform distribution of the lost frames (not presented). In fact, when there is no adaptation the following example situation can happen: a sequence of 6 frames is fully presented but none of the following 4 will be. When the algorithm is working there is a smoothing effect: the algorithm reduces the frame rate and there is a stable presentation of frames (for instance, every 2 out of 3 frames).

An objective analysis of the lost frames can be made comparing some zones in the two previous figures. Notice for example that between t=75s and t=100s we had almost the best quality in the first experiment with low losses, whereas in the second there were more losses (around 10 to 15% of the frames). Another great difference can be noticed after the great reduction of bandwidth in the middle of the figures. In the first experiment, after t=220s the best quality with low losses was reached while in the second experiment losses decreased quicker but they remained between 10% and 15% throughout the experiment.

5.2 Experiment 2: Comparison of the QoS with and without considering the end host situation

The objective of the second experiment was to analyze the relevance of the inclusion of the end hosts situation in the algorithm. Again, two similar tests were made. In the first the full algorithm was working (figure 5) whereas in the second only the network related information of the receiver reports was used to calculate the filtered losses (figure 6). The system was tested with a faster host sending information to a slower one and with the network bandwidth being reduced and increased as in 5.1.

Figures 5 and 6 show the frames not shown in each time interval. The first interval is when there was no limitation to the available bandwidth, in the second the bandwidth had a limit of 200 Kb/s, the third had 100 Kb/s, the fourth had again 200 Kb/s and finally the fifth had no limitation.

When the full algorithm was running, frames were not displayed due to two reasons: losses and the algorithm decision to reduce the number of frames. The two components can be seen in figure 5. We sum these two values in order to compare both situations.

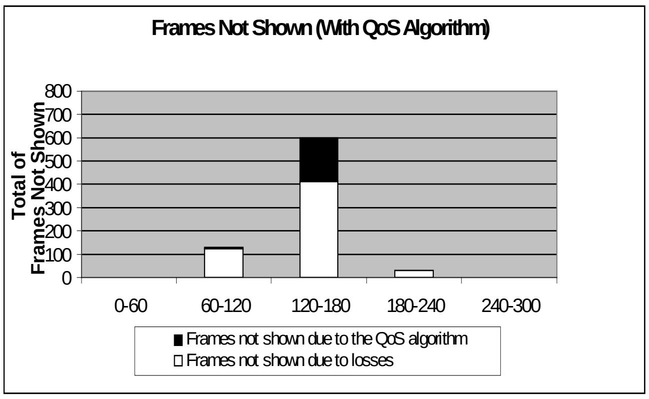

Figure 5. Experiment 2: Frames not shown with the full algorithm working.

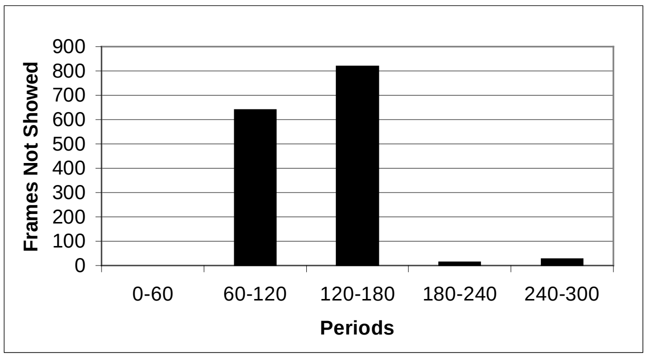

Figure 6. Experiment 2: Frames not shown without the end hosts part of the algorithm.

In fact, the number of frames not displayed in the situation with the full algorithm is much lower. As the frame rate values in all levels of the scale is almost similar, the number of frames presented with the full algorithm is much higher and so is the QoS.

The subjective analysis also proved the quality in the first case to be much better and less disturbing. This permits us to conclude the importance of the inclusion of the end hosts in the algorithm.

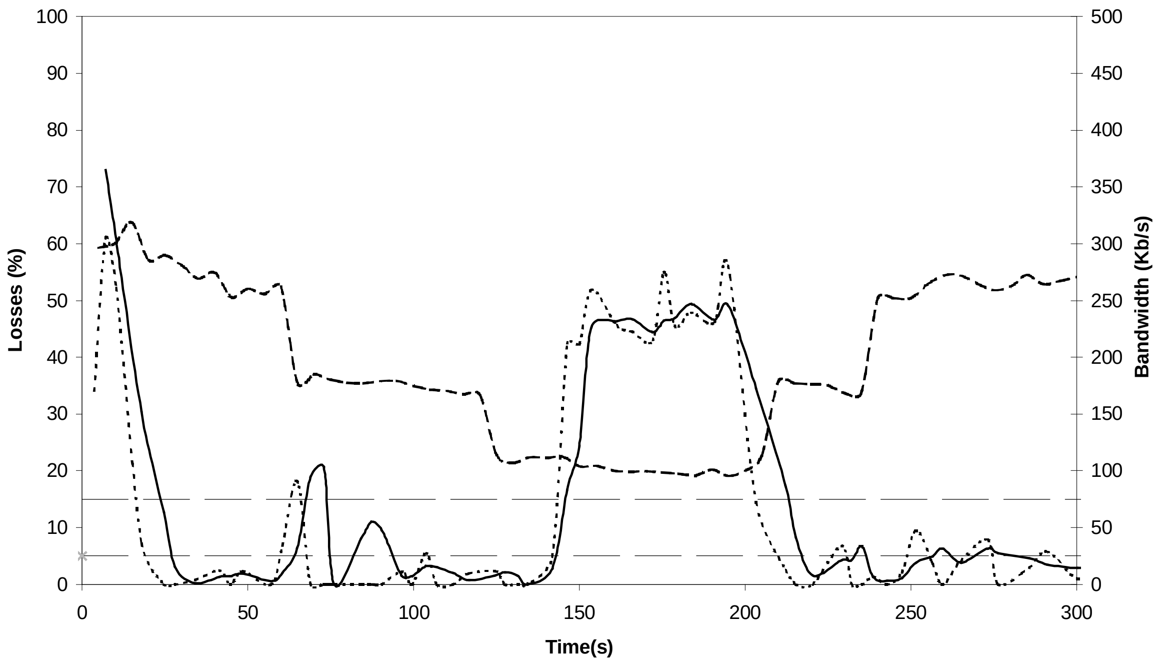

Figure 7. Experiment 1. Normal operation of the algorithm.

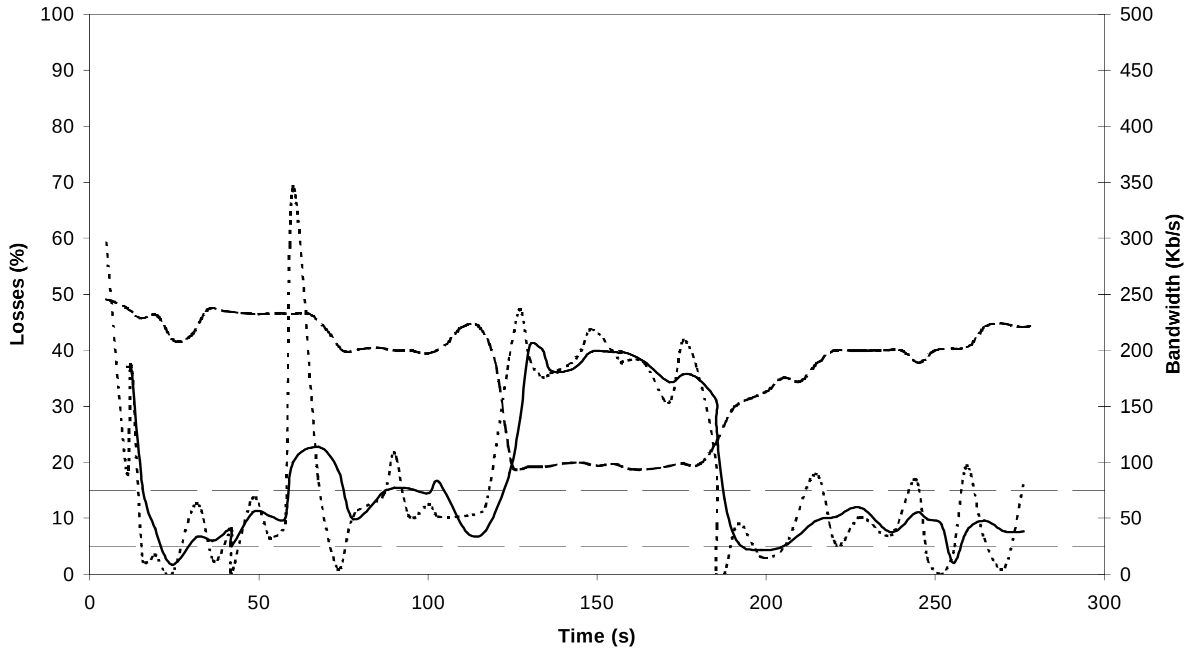

Figure 8. Experiment 1. Operation without QoS adaptation.

5.3 Experiment 3: Algorithm reaction times

The third experiment consisted on measuring the algorithm reaction times to a bandwidth reduction or increase, i.e., the time it takes to change of QoS level when a problem occurs.
The relevant values to analyze are: (1) the round-trip delay between the two end hosts; (2) the time between RTCP receiver reports; and (3) the number of receiver reports used to obtain the filtered losses.
The round-trip delay has no significance in our system because the network has only one switch and it is very fast. The minimum interval between RTCP repo
rts was 3 seconds and the formula from the standard was used to prevent peaks of reports. The measured average between reports was, in practice 5 seconds (the RTP standard defines this interval to be 5 seconds which means a bigger value after the formula is applied, but we have found the system too slow and reduced it). The filtered losses value was calculated with three reports (around 15s to obtain these 3 values). Even if the network was bigger with a longer transit delay, the effect of the other two values are decisive to the reaction time of the control loop (notice that for video-conference an usually accepted maximum value for the round-trip delay is 0.3 s).
Figures 9 and 10 show the distribution of the reaction times. More than 50% of the reactions took place in the first 10 seconds. Almost all took place before 20 seconds.

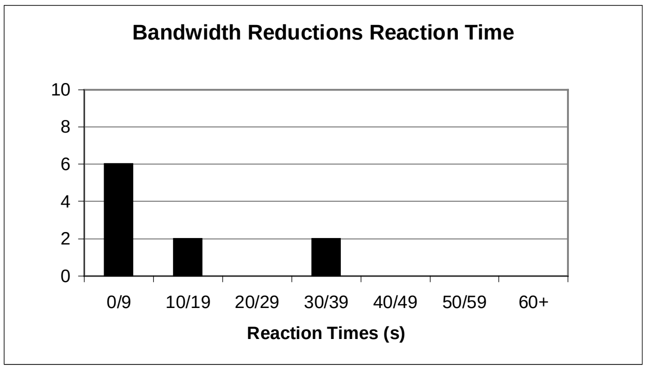

Figure 9. Experiment 3: Reaction time of the system to bandwidth reductions.

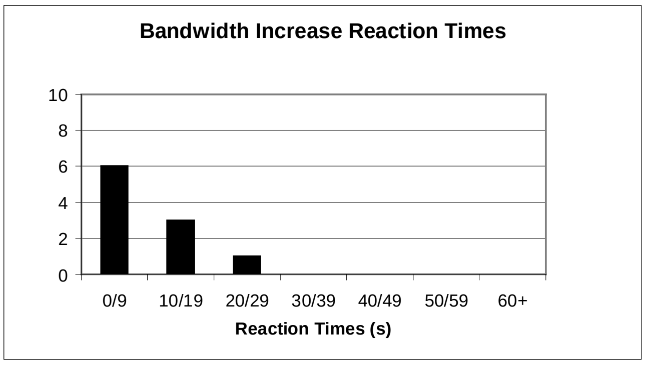

Figure 10. Experiment 3: Reaction time of the algorithm to bandwidth increases.

This was expected if we take into account the interval between RTCP receiver reports and the number of these packets to obtain the filtered losses average.
Shorter reactions can be obtained if the report intervals will be shortened. However, this increases the amount of bandwidth devoted to control aspects of the session.
One factor that must be considered and could not be tested is how a real ATM network behaves in regard to traffic that overpasses the contracts. There are no general statements about this because it is related with the traffic load at any moment. Nevertheless, we assumed that the process is relatively slow.

6. Related Work

Some similar work on QoS adaptation algorithms have been done already. [4] presents a QoS adaptation algorithm also based on RTP. However, the end host conditions are not considered; the algorithm works only with the packets lost value given by RTCP reports. [17] presents a very similar algorithm but also considers some problems related to multicast.
[5] presents a transport protocol for continuous media based on a similar algorithm. The algorithm, as ours, is not part of the application. They do not use RTCP for monitoring and the protocol proposed is part of a "Quality of Service Architecture" (QOS-A) [6].
Another adaptation scheme to control network congestion is presented in [18]. It is based in bit rate and packet rate scaling, i.e., they do not consider discrete QoS levels as in our *QoS scale*.
In [8] we proposed algorithms for intramedia and intermedia synchronization. We also proposed a QoS adaptation algorithm similar to the one presented here but simpler and without the framework. Our present objective is not just to solve the QoS problems but to simplify the programmer’s job in creating multimedia applications. The framework also enables the application to take the most appropriate decisions making the algorithm generic to every type of media.
ISO proposed a generic framework [10] in order to provide a basis for the development and enhancement of standards that specify QoS mechanisms or mention QoS requirements. It proposes a common terminology to be used by these standards. In some sense our work is an implementation of some of the concepts presented in this document.
A conferencing application based in RTP is presented in [12]. Our proof-of-concept videoconference is similar to theirs, although much simpler. They present many generic interesting issues related to the work presented here: RTP, video compression formats, ALF and ILP [7]; and also IP multicast.
A question that we did not studied but that is related to QoS adaptation is receiver-dependent QoS for multicast. This question is studied in [9]. The solution proposed is based on filters. "Quality Query by Example" can be implemented over our API but other solutions can be used

as well.
Another side question is the maximization of quality when MPEG is the compression format for video. This issue is analyzed in [11] and [14]. The idea it to schedule the stream in such a way that less priority frames (B and P) are the first candidates when the network has to discard information.
[19] is a survey about QoS in distributed multimedia systems. It lists several questions related to QoS and approaches to solve them. It refers the necessity of hiding the internal system QoS parameters from the user. They mention "Quality Query by Example" as a solution to that problem. We also felt that necessity but used a more powerful solution that can be managed by the programmer at his will. The user meaningful concepts are then translated to system level ones.
Another type of translation between QoS parameters of different levels is shown in [13]. The paper presents a "QoS Broker" that negotiates QoS values between the application and the different system components. However, most of the parameters are related with real-time operating systems aspects which make the solution not so general.

7. Conclusions

This paper presents a framework for the programming of distributed continuous media applications like videoconference, video-on-demand or tele-presence.
The framework has two main components that handle most of the issues related to the continuous media processing: The Media Device Managers handle the media capture, communication and presentation for the application, and are media specific. The QoS Manager controls the QoS of the media streams, in a uniform and media independent way.
The framework adapts the QoS of the media to the network congestion level and end hosts load, under a higher or lower supervision by the application. This adaptation is made by an algorithm proposed in the paper, but others could be devised. The particular case of the proposed algorithm is not new in the literature. Our contribution is the inclusion of the end hosts load that we found in practice to be relevant. In fact, when using an ATM LAN it proved to be the real restriction to the media quality, not the network bandwidth. Another contribution is a mechanism to inform the QoS Manager of the various working levels the application is interested on having.
The complete framework provides a suitable solution for the new generation of applications that have to control the semantics of their data externally to the network. As the experiments showed, a uncontrolled situation produces an erratic behaviour that misuses the network and is more disturbing to the users.
The concrete value of the intervals for the reports, and the number of reports needed to produce the average can influence the performance of the closed control loop. We found that the standard value proposed in the RTP standard, and an average of the last three reports lead to a satisfactory system.

8. Bibliography

[1] Antunes N., Rocha R., Pinto P., “Analysis and Simulation of a Traffic Management Control Scheme for ATM Switches with Loose Commitments”, Int. Conf. On Networks and Distributed Systems Modeling and Simulation, Phoenix, 1997, ftp://mariel.inesc.pt/pub/papers/cndsmsc97.ps.gz

[2] Bom J., Marques P., Correia M., Pinto P., "An Architecture for Dynamic Multimedia QoS Control", 7th IFIP/ICCC Conf. on Information Networks and Data Communications, Aveiro, June 1998

[3] Bom J., Marques P., Correia M., Pinto P., "Integrated Dynamic QoS Control for Multimedia Applications", Int'l Symposium SYBEN 98 Broadband European Networks, Zürich, May 1998

[4] Busse I., Deffner B., Schulzrinne H., “Dynamic QoS Control of Multimedia Applications based on RTP”, Computer Communications, Vol. 19, Number 1, Jan. 96

[5] Campbell A., Coulson G., “A QoS Adaptive Transport System: Design, Implementation and Experience”, ACM Multimedia´96, Boston, 1996, 117-127

[6] Campbell A., Coulson G., Hutchison D., “A Quality of Service Architecture”, ACM SIGCOMM 94, Computer Communication Review, Vol.24, April 1994, 6-27

[7] Clark D., Tennenhouse D., “Architectural Considerations for a New Generation of Protocols”, ACM SIGCOMM 90, Philadelphia, 1990, 200-208

[8] Correia M., Pinto P., “Low-Level Multimedia Synchronization Algorithms on Broadband Networks”, ACM Multimedia´95, San Francisco, 1995, 423-434, ftp://mariel.inesc.pt/pub/papers/mm95.ps.gz

[9] Garcia F., Hutchison D., Mauthe A., Yeadon N., “QoS Support for Distributed Multimedia Applications”, Proceed Int. Conf. in Distributed Processing (ICDP´96), Dresden, 1996

[10] ISO/IEC JTC1/SC21, “Information Technology – Quality of Service Framework – Final CD”, July 1995

[11] Kalkbrenner G. et al., "Quality of Service (QOS) in Distributed Hypermedia Systems", Proc. 2nd Int´l Workshop on Principles of Document Processing, 1994.

[12] McCanne S., Jacobson V., “vic: A flexible framework for packet video“, ACM Multimedia'95, S. Francisco, 1995

[13] Nahrstedt K., Smith J., ″The QoS Broker″, IEEE Multimedia, Spring 1995

[14] Riley M., Richardson E., “Minimizing the Effect of Cell Losses on MPEG Video”, BRIS´94, Hamburg, 1994, 491-494

[15] Schulzrinne H., “RTP Profile for Audio and Video Conferences with Minimal Control“, (RFC 1890) January 1996

[16] Schulzrinne H., Casner S., Frederick R., Jacobson V., “RTP: A Transport Protocol for Real-Time Applications”, (RFC 1889) January 1996

[17] Sisalem D., “End-To-End Quality of Service Control Using Adaptive Applications“, IFIP 5th Inter. Workshop on QoS, New York, May 1997

[18] Talley L., Jeffay K., ″Two-Dimensional Scaling Techniques for Adaptive, Rate-Based Transmission Control of Live Audio and Video Streams″, ACM Multimedia´94, S. Francisco, 1994

[19] Vogel A., Kerhervé B., Bochmann G., Gecsei J., “Distributed Multimedia and QoS: A Survey”, IEEE Multimedia, Vol.2, Numb2, 1995