Integrated Dynamic QoS Control for Multimedia Applications

João Bom[b], Paulo Marques[b], Miguel Correia[c], Paulo Pinto[a,b]

[a]Instituto Superior Técnico, Av. Rovisco Pais, P-1000 Lisboa Portugal
[b]INESC, R. Alves Redol, 9 P-1000 Lisboa, Portugal
[c]Universidade de Lisboa, Faculdade de Ciências, Campo Grande, Bloco C5, Piso 1, 1700 Lisboa[*]

ABSTRACT

This paper presents an algorithm to control QoS for distributed multimedia applications. The algorithm approaches the problem in an integrated way solving both the problems of the network bandwidth and the load at the workstations. It interacts with the applications by an abstract sequence to describe values for QoS that are more meaningful to application programmers than the low-level entities such as cell loss rate, throughput, etc. Some experimental results of the algorithm are presented.

Keywords: Quality of Service, Distributed Multimedia Applications, Real-Time Transport, Adaptation Algorithms

1. INTRODUCTION

Distributed multimedia applications start already to appear in our working (and leisure) environments. However, they still use highly homogeneous networks, equipment and encoding algorithms. It is not hard to predict that this restriction will soon be overcome, but it is important to assure that the generalization will not be intrinsic to the applications, in order to keep them simple and easy to build. I.e., the core of the application (the functional part) must be shielded from the problems of adaptation to networks, or the evolution on encoding algorithms. Ideally, a multimedia application can grab as much machine power or network bandwidth as it can get. Therefore, the core must be programmed as if these resources were infinite and have adjusting mechanisms performed externally and in a media, and network, independent way to indicate a limit somewhere. Then, applications will work with the maximum conditions they can get at the moment in terms of operating systems and networks, and will be prepared to survive technological advances.

The concept of Quality of Service (QoS) can play a major role in the overall process involving the application and the adjusting mechanisms because it can be made abstract enough to handle different kinds of problems. However, it is important to find: (a) application meaningful concepts for the parameters that influence the performance (typically, bit error rate, or call establishment latency are of little relevance to applications); and (b) the dynamics of the adjusting mechanism to always get the best of what the user is willing to have.

This paper is concerned with distributed multimedia applications using broadband networks, and with the control of the QoS when the dynamic conditions of the connection can vary substantially. A basic aspect of QoS when the network conditions are fairly stable was already done[6] and it covered the maintenance of low-level intramedia and intermedia synchronization for continuous streams. Basically, it was important to control jitter and skew, and detect temporary network congestion or machine loads. However, at such low-level it is difficult to decide what are the best choices when the operating conditions change greatly. Specially because they are very application dependent.

New multimedia applications produce variable bit rates due to various reasons (the algorithms themselves, or extra techniques such as silence suppression on audio), so they fit well on variable bit rate network services. The ATM technology provides a good networking solution for these requirements but creates a scenario where dynamic operating conditions can happen easily. When setting up a connection the reservation of the maximum bandwidth is too costly and corresponds to a bad use of the network resources. Most of the times, the connection traffic parameters are set to slightly greater values than the ones produced by the encoders. This creates a safety margin to prevent anomalies. However, most of the Call Admission

---

[*] {joao.bom,paulo.marques,paulo.pinto}@inesc.pt mpc@di.fc.ul.pt

Control algorithms are conservative[1] and some network resources are just never used. Depending on the UPC mechanism at the network interface, users can overpass the contracts of the connections and use network resources without a guarantee.

If the application wishes to exploit the statistical nature of the network and tries to improve the quality of service by going over the connection traffic contract, the operating conditions will even get more dynamic. Not all applications can behave this way, of course. They have to handle most of the situations they can incur by using non-guaranteed conditions (errors, loss of synchronism, etc.). The point is that this exploitation of unreserved network resources can use the same technique of the adjusting mechanism introduced above.

This paper describes an integrated application-transport control algorithm to adapt the application and take advantage of these spare resources. The transport part of the algorithm is in some sense transparent to the application. The application simply defines the possible values of QoS each media can take and the events that it wants to have knowledge of (for example, minimum video quality reached).

The algorithm changes the bandwidth used by the application by changing the quality of one or more of the media being sent. If, on the other hand, an application repeatedly tried to process more data than the available bandwidth and end systems capacity the overall performance would be lower than what it could be if the requirements were more realistic. That is the reason why the amount of data being sent, and in consequence the quality of the media, should be adjusted to the capacity of the overall system.

The novel parts are that the algorithm takes into account in its control loop the operating conditions on both machines, and not only the network situation; and that it operates based on an abstract range of severity levels making it suitable to adapt to various types of media encoding. I.e., applications provide a range of different acceptable operating conditions together with their encoding dependent solutions to the variations of these conditions, in order to enable the algorithm to exercise its control autonomously (as long as it stays within the range).

2. ARCHITECTURE

This section discusses some issues related to transport protocols for continuous media and ATM network technology.

2.1. Transport protocol

Traditional transport protocols, such as TCP, have already too much assumptions in relation to the semantics of the data and the network. TCP is not suitable for continuous media for several reasons: it imposes its own transmission rhythm to the media being sent when it should be the application doing so; it imposes the correct deliverance of the data making use of retransmission but it could be acceptable to tolerate some losses but no retransmissions; the congestion control acts drastically when a missing segment is noticed never assuming that an error could have occurred. On the other hand these protocols work and make decisions on abstract portions of data (data units) with little relevance to the application.

New generation transport protocols, such as RTP[9], outcome most of the previous limitations: they are based on the concept of Application Level Framing[5] (ALF) working on data units that are relevant to the applications, and integrate their processing with the application own processing[5] (Integrated Layer Processing). Moreover, RTP has an associated control protocol -- RTCP -- to assess the network conditions but does not impose any pre-defined algorithm to work on the control data. RTCP basically sends reports from the sender to the receiver and vice-versa. These reports have information about the packet loss, received and transmitted rates and delay jitter, that can be used by the applications to take decisions.

We used the RTP protocol in our experiments but it is more important to highlight these characteristics as being vital to the transport layers of multimedia applications than to study the particular case of RTP. In fact, we felt the need to add some information to the RTCP control packets. The standard information is simply related with the network and it is also relevant for the QoS control to have some knowledge of the local behaviour of the applications (specially as seen by the receiver machine).

2.2. ATM

It was stated at the introduction that the application will use non-guaranteed network resources, as a means to improve quality without the expenses of a guaranteed scenario. One alternative is to use the Available Bit Rate (ABR) class of service of ATM. However, there are several reasons why this is not suitable for multimedia: the ABR was designed to serve data applications that can use spare bandwidth. Data applications can typically adapt to varying conditions on the network but an important factor is that cell loss should be as low as possible (data applications are rather sensitive to information loss). Therefore, ABR has its own control cycle to govern bandwidth and prevent loss situations. The multimedia case is different: the adaptation to different bandwidth conditions is a slow process; some losses can be tolerated in the process, although, of course, they are not desirable; and the ABR control loop will behave strangely to multimedia because it is based on cells which are not a multimedia abstract concept (the same problem as the approach of traditional transport protocols described above).

The natural choice is to use Variable Bit Rate (VBR) class of service, and use priorities to work over contract values. The application sets a lower limit of bandwidth for the connection, below which it is useless to maintain the interaction. Then tries to raise the quality by using CLP=1 cells as much as the network can handle. We assume that the UPC/NPC mechanism uses cell tagging until the Peak Cell Rate (PCR) value and discards cells above this rate. Therefore, a video connection, for instance, should set the PCR in accordance to a frame length, but can set the Sustainable Cell Rate (SCR) lower than the actual value it intends to use (and just to cover the minimum limit).

3. END-TO-END CONTROL ALGORITHM

QoS in multimedia has a slightly different meaning than the traditional concept in the OSI model[8]. Although both end up to values and rates of cell loss, throughput, delay, etc., it is difficult to relate the relative importance of these entities to the overall multimedia quality. A multimedia application user has a subjective assessment of the data he is watching and listening to. Therefore, it is very difficult to get lower level information such as the one listed above from the user. The control unit should rely on higher level information and a scale, the range of operating levels explained above, fits this objective. The user can express his intentions in terms of video frame rates, quality factors, audio sampling rates and encoding quality, etc.

The control algorithm works in two main areas: QoS monitoring and QoS control. The aim of the first is to observe the working conditions, whereas the second acts to adjust the system towards a new stable condition when problems happen or disappear.

This section describes our proposes to handle these two issues: QoS monitoring (3.1) and QoS control (3.2).

3.1. QoS monitoring

QoS monitoring works as an end-to-end closed control loop based on the RTCP reports. Both the sender and the receiver should send RTCP reports but only the reports sent by the receiver are considered. So, the algorithm is executed exclusively in the sender.

The algorithm is cyclic and reacts each time an RCTP receiver report packet is received by the sender (see figure 1). Monitoring looks both at the network level QoS parameters (e.g., cell loss, jitter) and application level QoS parameters (e.g., number of discarded frames at the receiver due to machine overload). In order to have the latter information, the control packets had to be extended with the following two fields (the extension feature was considered in RTCP by the inclusion of *profile-specific extensions*):

- **too_late** – indicates the number of frames dropped by the receiver due to a delay that prevented them to arrive before their presentation time.
- **nshown** – indicates the amount of frames discarded by the receiver due to machine overload, i.e., frames that were received in time to be presented but that were not presented because the process that should perform that action did not get active in time to do it. This is a consequence of not using real time operating systems.

All these indicators must be assessed in relation to the sampling rate. Obviously, it is more relevant to lose one frame at 5 fps than to lose one at 20 fps, in the video case. The sampling rate, as it will be described in the next section, can vary during the connection time.

3.2. QoS control

The QoS control algorithm works at three stages: the first two more autonomous and the other more guided by the application.

The first stage is used as a first approximation to a problem. The idea is to assume that the problem is transient and some data must be dropped. The application provides indications about the semantics to drop data (for instance, an M-JPEG stream can drop video frames; an MPEG stream can drop small amounts of data (B frames), larger amounts (P frames and B frames), or large amounts corresponding to a complete sequence (P frames including the first I frame)). The second stage is used when the first one does not produce results, and the algorithm goes up and down the scale provided by the application.

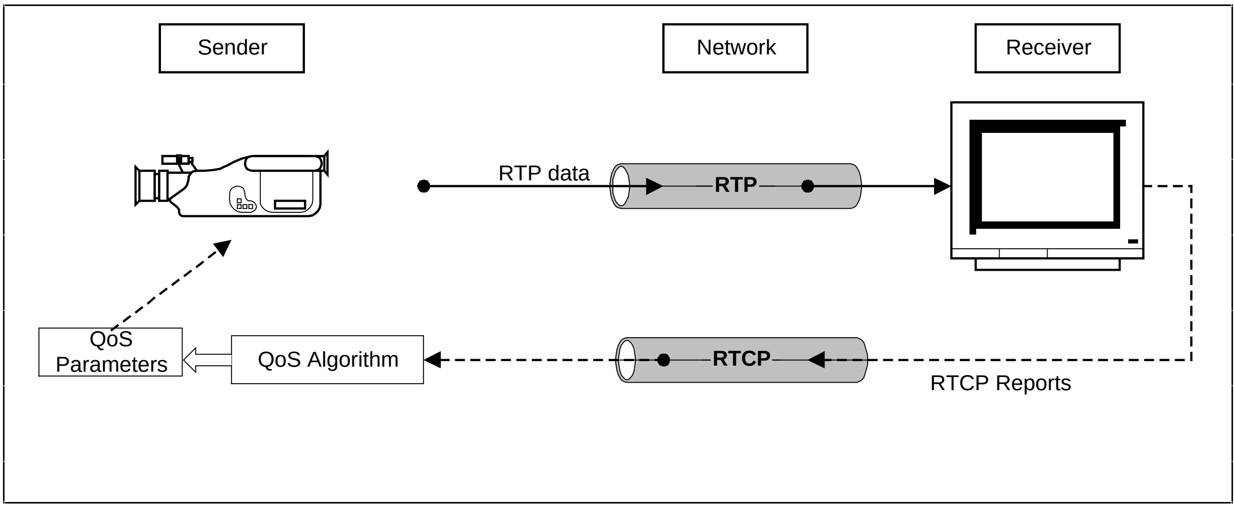

Figure 1 - QoS Monitoring Cycle

The third stage is entered when these adjusts do not solve the problem and some major decision has to be taken. Essentially the lowest level of the scale was reached and the problems persisted. The application is informed and decides what to do. For example, it can terminate the connection or submit a new, less demanding scale for QoS.

As the third level is more trivial, the rest of the session is devoted to the first two stages.

The application has to inform the algorithm about the values of QoS the media can take, and its desired initial conditions. These values are given as a *scale* of QoS parameters with a number of *levels*. Each level is expressed in terms of values of given QoS parameters. For example, a very simple video QoS scale can be expressed in terms of the frame rate and quality factor (Q) - figure 2.

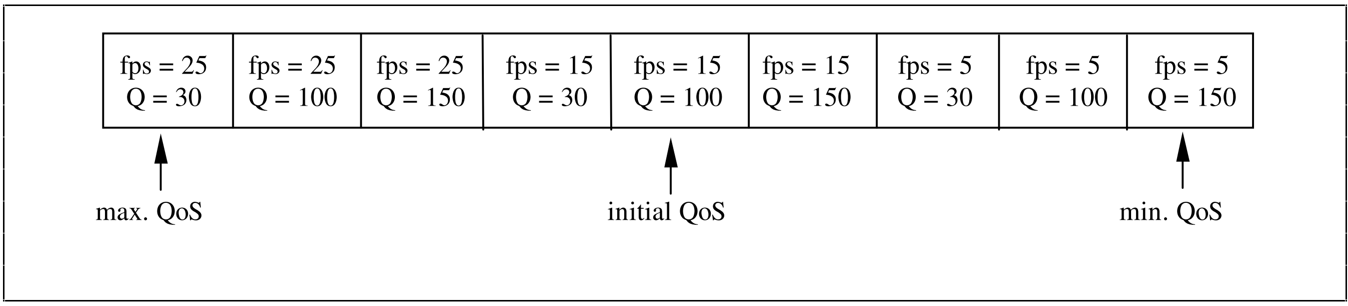

Figure 2. A QoS scale for a video stream.

The QoS control can change the QoS working level over that scale. It starts at the initial level and can move downwards or upwards according to its assessment of the working conditions (bandwidth on the network and CPU loads at the machines involved). These decisions of moving from one level to another will be considered next.

The control algorithm gets values over which it acts from the RTCP reports. To avoid the influence of possibly erroneous values and to limit the oscillations on the algorithm, an average of the last values is performed. In the experiments we used the three past values of the reports for the average. We used the sum of three values -- too_late, nshown and the frames lost in the network – to calculate the average. This last one is obtained from the number of packets lost, that comes in the standard part of the RTCP reports. The sum is divided by the length of the period between RTCP reports. This last value is the one used to calculate the average. It is a value of the (total) losses per second.

The average value can fall in one of three zones on a scale (see figure 3): the degradation zone (between λ_s and 100%); working zone (between λ_i and λ_s) and improvement zone (between 0 % and λ_i). The values λ_i and λ_s are calculated experimentally and depend on the data type (media) used.

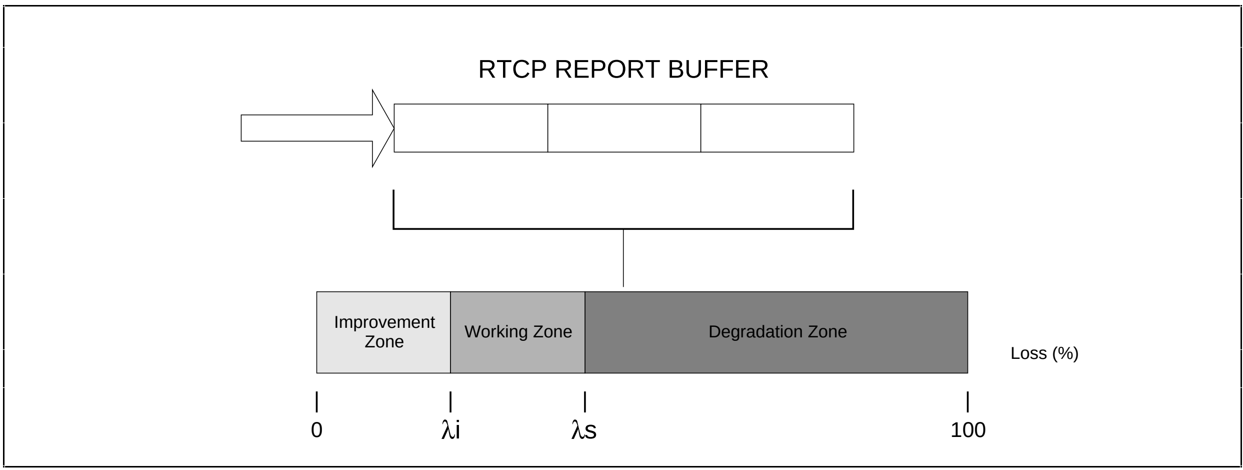

Figure 3 – QoS Control Algorithm

The control algorithm works as follows:

- When the average values fall into the working zone the algorithm acts autonomously. Inside this zone it is assumed that a reduced part of the information is being continuously lost. For example, for video, discarding a frame once in a while -- first stage of the control algorithm. These losses are not noticeable to the users and constitutes the stable stage of the algorithm.

- When the average values fall into the degradation zone the algorithm moves downwards on the QoS scale provided by the application to reduce the volume of data. When the system enters into this zone the conditions changed in such a way that a considerable part of the information has been lost.
- When the average values fall into the improvement zone the algorithm moves upwards on the sequence to try to use more bandwidth, as the network seems to be accepting the values at the moment. Basically, there has not been residual losses for a while.

The degradation and the improvement of QoS are not symmetrical in the following sense. A degradation is performed when there is not enough bandwidth in the network or when there is an excessive CPU load in a machine. The improvement is not performed in the contrary situation but when degradation situation stopped to be detected. Determining if there was, for instance, more bandwidth available would require a specific mechanism that would cause extra load in the machines and use more bandwidth in the network.

The algorithm itself is only aware of losses at the application level, i.e., it does not distinguish between network losses and local losses due to excessive CPU load in a machine. The reason for this is that the algorithm itself does not have to know the causes of the losses, it only has to react to them in a way that allows the application to work, even with less than the initial required QoS.

3.3. An implementation of the algorithms for MJPEG video

An implementation of the algorithms was executed as part of a videoconference prototype. The algorithms monitor and manipulate video transmitted in MJPEG format. The two QoS parameters considered for the QoS scale were the sampling rate and the JPEG quality factor. Figure 3 shows the nine levels defined and used for the experiments. Reducing the QoS is equivalent to move to the right in the scale. As a principle, the first degradation is the reduction of the sampling rate and then, the Q factor. This precedence is based on the fact that changes on the Q factor introduce greater load on the machines than changes on the sampling rate.

1	2	3	4	5	6	7	8	9
fps = 25 Q = 50	fps = 22 Q = 50	fps = 19 Q = 50	fps = 25 Q = 75	fps = 22 Q = 75	fps = 19 Q = 75	fps = 25 Q = 100	fps = 22 Q = 100	fps = 19 Q = 100
				initial QoS				

Figure 4. QoS scales used in the experiments.

To improve performance, both the receiver and the sender must be aware, as soon as possible, of the dynamic changes in the QoS parameters introduced by the QoS control algorithm. This can be done in different ways, but our choice was to have an explicit indication in every data packet. The decision was made in order to avoid a significant increase in the overall load across the monitoring cycle. It became obvious that it would be better to waste a few bytes, increasing slightly the bandwidth used, than to run an algorithm on both the sender and the receiver sides (using the timestamp, etc.) to deduce the sampling rate.

The indication consists of the sampling rate because the Q factor is already included in the RTP header extension for MJPEG. This bandwidth increase can be considered residual compared with the bulk of the multimedia data. The knowledge of the sampling rate by the QoS control algorithm at the sender is important because the above indicators (nshown and too_late) must be assessed relatively to it (as it was already stated, it is obviously more relevant to lose one frame at 5 fps than to lose one at 20 fps). Each indicator is evaluated relatively to the sampling rate using the proportion: 1 / *Sampling rate*. On the receiver's side the value for each indicator is computed between the sending of two RTCP packets. On the sender's side, each time an RTCP receiver report packet is received, the value of the indicators must be converted to a base unit in order to compare these values across the cycles with potentially different sampling rates. The base unit will be the percentage of loss per second.

4. EXPERIMENTS

This section describes three experiments and their respective results. The goal of the experiments is to demonstrate the algorithm under different network and local machine conditions. The experiments consisted on video images with movement and some background detail, transmitted from one workstation to another. The video is encoded in JPEG using Parallax boards. The workstations (Sun Sparc10) are connected by an ATM switch. The threshold values chosen for the experiments were $\lambda_i = 5\%$ and $\lambda_s = 15\%$.

The RTP standard proposes an algorithm to calculate the interval between consecutive RTCP packets. This algorithm was designed considering that control traffic should be a small and known fraction of the total used bandwidth. The algorithm is based on the following principles:

- the interval between RTCP packets must be greater than a minimum time (5 seconds was the value established by the standard) to avoid having bursts of RTCP packets;
- to automatically adapt to changes in the amount of control information, an estimate of the average RTCP packet size is calculated and considered in the algorithm.

This interval has a direct consequence on our control loop. We decided to lower the minimum interval to three seconds. With a value as high as the one established by the standard, the algorithm took too much time to make a decision. It had to wait at least five seconds to receive a RTCP packet and then make a decision. If the conditions change abruptly, then three reports have to be received. With such a large interval when the packet arrives the state of the network or the state of the local machine could have changed, so the practical effects of the algorithm could have been the contrary. Our minimum value of three seconds was experimentally obtained, and it revealed adequate to the algorithm’s goals.

The RTP was implemented as a user process. There are some inefficiencies due to this option. However, the less rich results in terms of bandwidth used were not relevant for the purpose of this paper, and can even be considered a good example of a limitation at the end system workstation.

4.1. First experiment

The purpose of the first experiment (figure 5) was to study the effect of the machine loads on the overall efficiency of the system. No restrictions were imposed on the network bandwidth and the machines had a constant load throughout the experiment. We have chosen a unusually large video image and it was needed at least two packets to transport a video frame. Therefore, a sampling rate of over 20 frames per second is a very high load to the workstations. The figures of all experiments show the bandwidth really used (higher line) and the filtered (averaged) losses (solid line).

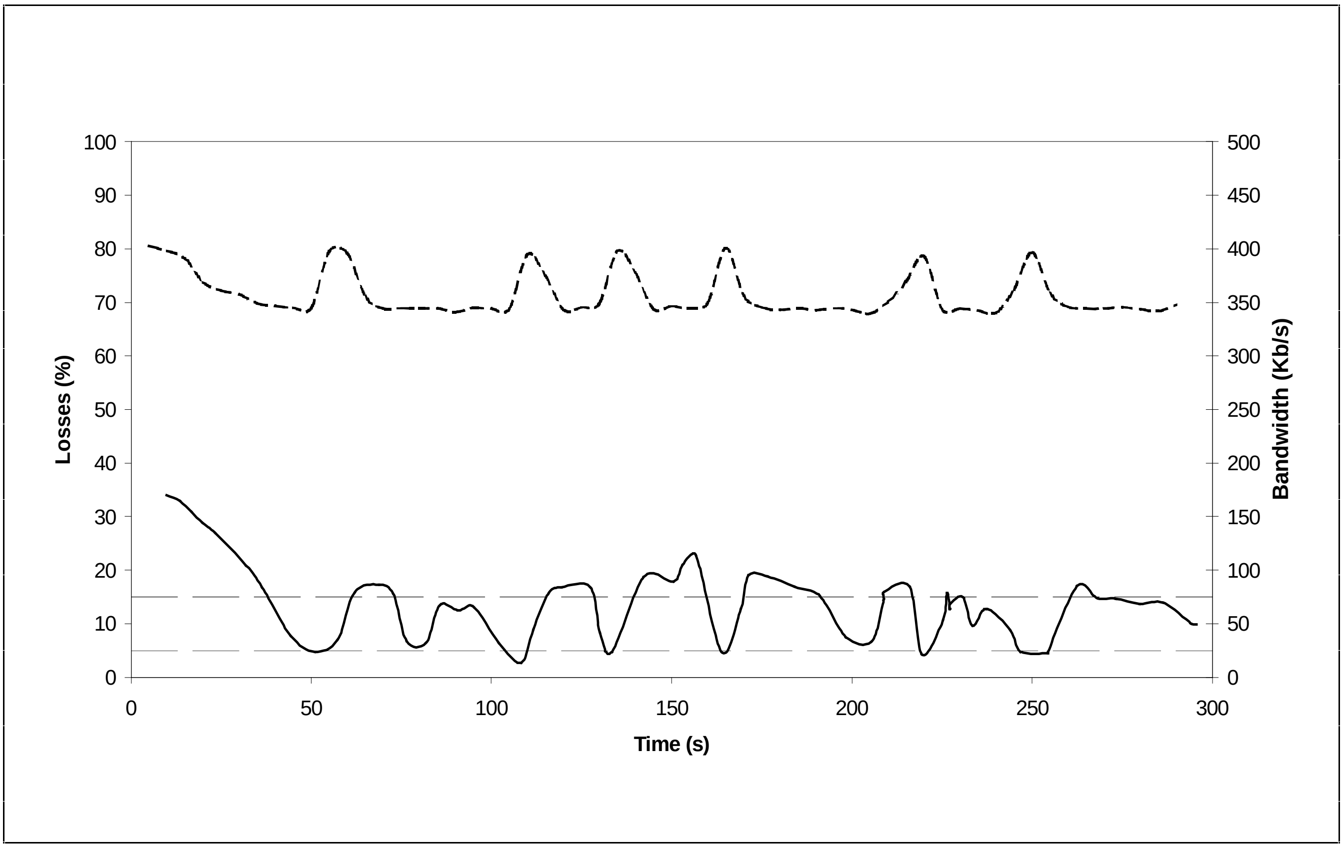

Figure 5. Results of the first experiment.

The initial level required for the QoS was too high for the system to cope with and losses started to happen. As a result the QoS level was reduced to the lowest level possible (level nine). At t=35s the system would have gone to an even lower QoS level than level nine meaning that the third stage of the algorithm would have happened. This part of the algorithm was not implemented, so the system remained at the lowest level. Anyway, the adaptation process proved to be too quick and at t=50s the system moved to level eight increasing the QoS. Losses started again to increase and the system moved again to level nine. The rest of the experiment is a sequence of these attempts as the correct QoS for these conditions seemed to be somewhere between level eight and nine. Each time there was a QoS increase the bandwidth increased but the receiver (a slower machine) could not cope with the pace.

4.2. Second experiment

The second experiment had artificial bandwidth cuts introduced on the network. The image size for this experiment and the next one was smaller than the one of the first experiment. The experiment begins without any limitation on the bandwidth, and at t=120s the bandwidth was reduced to 300 Kbps. After a while, at t=180s there was another reduction to 250Kbps and another one at t=260s to 200Kbps. After this the bandwidth started to increase to 300Kbps at t=360s and without limitations after t=420s. Figure 6 shows the filtered losses and the used bandwidth as a function of time. It is clear where the cuts and increases were performed.

The purpose of this experiment was to show how the algorithm worked if the required bandwidth (to support the required QoS level) was not available. It is possible to see how the algorithm adjusts and stabilizes under different operating conditions.

At the start of the experiment, the initial QoS chosen by the application was clearly less than the current conditions of the system. The losses went down the lower limit, and there was an increase on the scale until it reached the highest level (level one) at t=30s. It could even have gone higher. When the first cut was applied losses started to happen and the control algorithm moved down the scale to stabilize at the fifth level. Before it got stable it went down to the sixth level and returned to the fifth (around t=160s). The second cut deteriorated the conditions just a little bit and the losses increased but not very much. The algorithm needed not go down the scale during this phase. When the third cut happened, then a sudden increase on the losses took place and the algorithm had to go down the scale to level nine. Once again, the algorithm would have gone lower if it could to try to solve the loss problem quickly. This case is curious because it proved that level nine was the appropriated one and losses start to go down after a while. At t=290 losses went down the threshold, the algorithm stabilized and was even near to try to improve a little bit at t=340. At t=360s it started to improve the QoS level and went until level one.

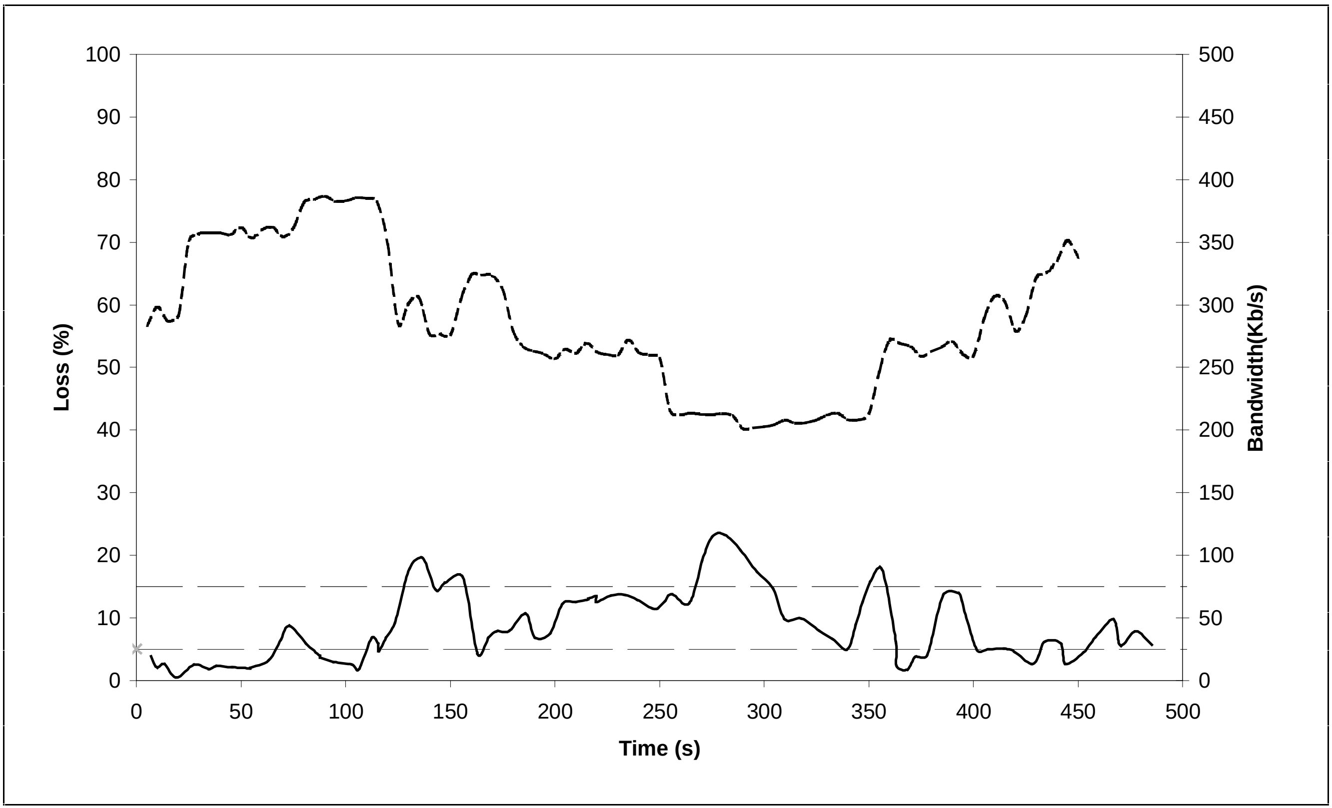

Figure 6. Results of the second experiment.

4.3 Third experiment

The third experiment had also artificial cuts on the network bandwidth, but the idea was to cut it drastically and then it increased and decreased with time. The experiment begins without any limitation on the bandwidth, and at t=100s the bandwidth was reduced to 150 Kbps. After a while, at t=200s the bandwidth started to increase to 300Kbps and after that to 400 Kbps at t=300s. There was a final cut to 300Kbps at t=350s.

Just like in the previous experiment the initial level was too conservative and an improvement was performed. Actually there was some disturbance at the beginning and the algorithm went to the sixth level at t=15s before it climbed to the third level at t=30s (this was due to the arrival of the first RCTP packets). It remained there until t=100s. The stable level was a little bit too high and the losses increased with time. If the t=100s cut did not happen, probably a correction would have taken place to go to the previous level (fourth level). As the cut happened, the system went down until it reached the level nine. Again, the system would have gone to a lower QoS level than this. As this did not happen and the conditions this time were adverse the losses started to increase. When the bandwidth was increased, the system started to raise in the scale until it reached the top level (specially because a new increase on the bandwidth was performed). It went very quiclky to level four between t=200s and t=220s with some disturbances around t=240s and down again to level one just after t=250s. The losses went down and the system could even go higher in QoS if the scale allowed it. After t=350s the algorithm started to decrease the QoS as the bandwidth started not to be available.

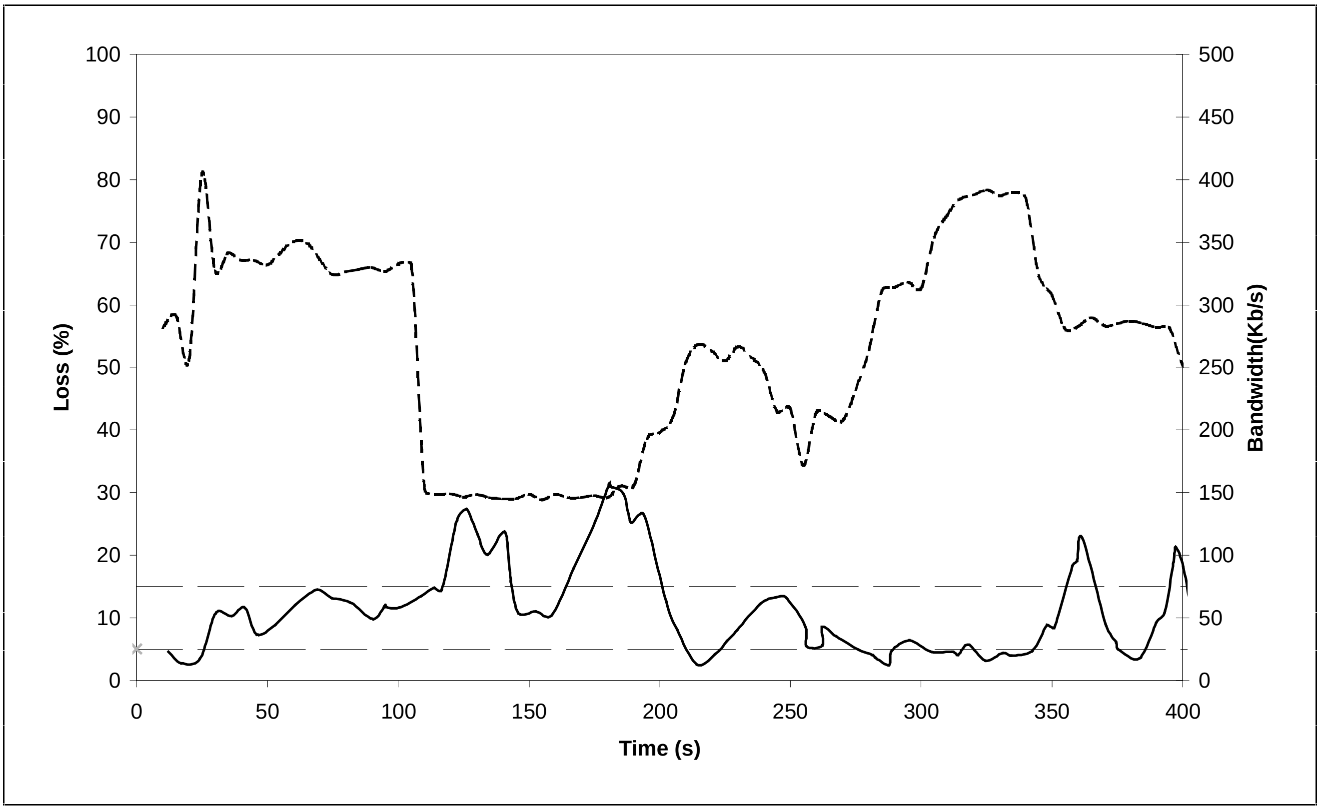

Figure 7. Results of the third experiment.

5. RELATED WORK

This section compares some works to our own. The works more similar to ours are [2],[3] and [14].

Busse et al.[2] presents a dynamic QoS control mechanism for continuous media based on RTP that is similar to ours. Some differences exist though. It just considers the congestion in the network but we also deal with the loads in the machines involved. It decides about the network conditions based only in the losses reported by RTCP received reports; we added two fields to these packets in order to understand not only the machines statuses but also to have a better perception of the network state (we get also information about losses due to excessive delay). A last difference is that [2] considers the algorithm as part of the application and we consider it as part of a software layer that handles the complexity of QoS control for the application, although very integrated with the application.

Another work very similar to ours is [10]. It has the same similarities and differences in relation to our work referred above about [2]. It considers also scalability issues related to the use of multicast.

Campbell[3] also presents a work similar to ours. They also consider a layer, the "multimedia enhanced transport layer", that handles the control of the QoS for the application. The API is described with detail. RTP is not used; monitoring information is passed using a specific mechanism. The transport system proposed is part of the "Quality of Service Architecture" (QOS-A) proposed in[4].

A framework for QoS management (or control) in a multimedia distributed system is given in[8]. It defines concepts and proposes a set of mechanisms that can be used to perform that management. Our work can be considered as an implementation of that general framework, although we do not use the same terminology.

A problem related but not treated by our paper is receiver-dependent QoS when using multicast. The problem is how to deliver different levels of QoS to different receivers in multicast conditions. This problem is focused in[7]. Filters are used to adjust the QoS of a stream to the specific receiver needs.

6. CONCLUSIONS

The QoS control for distributed multimedia applications has to take into consideration several aspects and only an integrated approach that includes the sender and the receiver machines can handle it entirely. This paper presented a closed loop control mechanism to manage QoS in such an integrated way.

Moreover, the applications can express their requirements using a framework that is more suitable to concepts that they are sensitive and not in terms of low-level QoS parameters. The application defines the "legal" QoS values in a *scale* composed of *levels* defined in terms of a set of QoS parameters. Coarse grain adjustments can be made if the extreme levels are reached.

Experiments have shown that the mapping between these concepts and the low-level ones can be achieved successfully and the subjective evaluation shown that users notice jumps on the quality of the session but it is not a major distracting factor.

A topic for further work is the use of this kind of algorithms in multicast configurations with special focus on scalability issues.

REFERENCES

1. N. Antunes, R. Rocha and P. Pinto, „Analysis and Simulation of a Traffic Management Control Scheme for ATM Switches with Loose Commitments”, *Inter. Conf. On Networks and Distributed Systems Modeling and Simulation*, Phoenix, 1997
2. I. Busse, B. Deffner and H. Schulzrinne, „Dynamic QoS Control of Multimedia Applications based on RTP”, *Computer Communications*,. **19**, Number 1, Jan. 96
3. A. Campbell and G. Coulson, „A QoS Adaptative Transport System: Design, Implementation and Experience”, *ACM Multimedia´96*, Boston, pp. 117-127, 1996
4. A. Campbell, G. Coulson and D. Hutchison, „A Quality of Service Architecture”, *ACM SIGCOMM 94*, *Computer Communication Review*, **24**, pp. 6-27, April 1994
5. D. Clark and D. Tennenhouse, „Architectural Considerations for a New Generation of Protocols”, *ACM SIGCOMM 90*, Philadelphia, pp. 200-208, 1990
6. M. Correia and P. Pinto, „Low-Level Multimedia Synchronization Algorithms on Broadband Networks”, *ACM Multimedia´95*, San Francisco, 423-434, 1995
7. F. Garcia, D. Hutchison, A. Mauthe and N. Yeadon, „QoS Support for Distributed Multimedia Applications”, *Proc. of the Inter. Conf. in Distributed Processing (ICDP´96)*, Dresden, 1996
8. ISO/IEC JTC1/SC21, „Information *Technology – Quality of Service Framework – Final CD*”, July 1995
9. H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, „RTP: A Transport Protocol for Real-Time Application”, RFC 1889, January 1996
10. D. Sisalem, „End-To-End Quality of Service Control Using Adaptive Applications", *IFIP 5th Inter. Workshop on QoS*, New York, May 1997