From a Master Server Architecture to a Feedback Control Architecture

Bechir Alaya, Claude Duvallet, Bruno Sadeg
LITIS, UFR des Sciences et Techniques
25 rue Philippe Lebon, BP 540
F-76058 Le Havre Cedex
email: Firstname.Lastname@univ-lehavre.fr

ABSTRACT

Today's multimedia applications ask for a complex set of quality of service (QoS) requirements. These applications define time, bandwidth and synchronization constraints and manage large quantities of data. Due to the similarities existing between multimedia applications and real-time database systems (RTDBSs), we propose, in this paper, an approach which consists in exploiting some works related to the QoS management in RTDBSs in order to apply them to multimedia systems. We particularly propose a FCS-MS[1] architecture, an improvement of the original master server architecture, which deals with QoS management by optimizing the resources use and reducing significantly the system overloads.

KEYWORDS
Distributed Multimedia Systems, Feedback Control Loop, Quality of Service, Real-Time Systems.

Introduction

These recent years, there are many researches which are not only interested to do minimize computation time, but also to satisfy application time constraints, i.e. deadlines, release times, etc. In this paper, we focus on a kind of these applications: multimedia applications. These applications must exchange very important quantities of data and they require the treatments to be done before fixed dates to guarantee an acceptable quality of service (QoS) in the streams presented to users. RTDBSs are the systems adapted to such data management while dealing with a certain QoS (Ramamritham, 1993) (Ramamritham et al., 2004).
Many distributed multimedia applications must face to predictable loads which cause the system overload. For example, user-demands may arrive burstly in a short period, which degrades the QoS provided to the user. Therefore, the need to design systems that provide a certain quality to users already served, has become a mandatory goal. In multimedia applications, the management of the QoS of the video packets allows to answer to these new needs. Since one decade, researchers try to adapt efficiently existing techniques to the video packets management without modifying the initial infrastructure. The main issue is the adaptation of the available resources (bandwidth, buffers size, video servers, etc.) and the proposition of news techniques to manage streams when instability periods (overload or underutilization) occur. The goal is to assure an acceptable QoS provided by the system to users, while respecting the multiple requirements of the video streams.
Several studies have focused on the definition of mechanisms and strategies which allow the system to provide an acceptable QoS.
The main architecture proposed for QoS management in the distributed multimedia systems is the FCS-MS architecture which is based on a feedback loop (Wu et al., 2001). This approach was inspired from the work on QoS management in RTDBS, due to the existing similarities between the data management constraints in distributed multimedia systems and those of RTDBSs (Amirijoo et al., 2006) (Kang et al., 2002a).
First, we present the multimedia system architecture that we use. Then, we present the new master server architecture we have developped to allow the QoS stabilization during the overload periods, more especially when the user-demands arrive in sudden manner. Finally, we present and comment the results simulation we have done to test the validity of our approach. We conclude this article by recalling the main ideas and by giving some perspectives to this work.

QoS in multimedia applications

QoS and distributed multimedia applications

QoS in a multimedia applications may be defined as the requirements in terms of bandwidth, quality of visualization, delay, and rate of video packets loss. Our approach consists in taking into account researches already done on the management of QoS in RTDBSs (Kang et al., 2002b) (Amirijoo et al., 2003) and their adaptation to multimedia systems. To this purpose, we propose an adaptation of a method based on feedback control architecture to distributed multimedia systems (Dulgheru, 2004).

[1] Feedback Control Scheduling for Multimedia Systems.

This adapted method is called FCA-DMS (Feedback Control Architecture for Distributed Multimedia Systems). Our approach consists in controlling the multimedia system congestion by discarding or not some clients requests of certain types according to the system state, notably to the video server capacity. The simulations we have done showed that this approach improves the QoS provided to users.

Feedback control architecture for QoS management

In a previous work, N. Dulgheru has proposed an architecture, named QMPEGv2 (Dulgheru, 2004) (ISO/IEC 13818-2, 1994) (Ng et al., 2000) which deals with distributed multimedia systems (cf. figure 1). The architecture proposed is composed of three main parts:

- ***A master server:*** It accepts requests from clients, chooses the video servers able to serve the demand, supervises the system state and adjusts the video streams in order to maintain the QoS initially fixed.
- ***Video servers:*** they send the video packets to the clients under the master server control.
- ***Clients:*** they send requests to the master server and receive the video frames from the video server. When a state change occurs, they send a feedback report to the master server.

When a video on demand is requested, the following steps are executed according to FCA-DMS architecture:

1. A client sends a request to the master server to get a video with a certain level of QoS;
2. The master server broadcasts the request to the video servers available in the system;
3. The video servers send back their responses to the master server, which chooses one among them;
4. A stream is opened between the chosen video server and the concerned client;
5. The master server asks the video servers to adapt their QoS, when necessary.

The feedback loop is used when there is a need to adapt the QoS to the load system conditions: it observes the QoS obtained by the client and, if necessary, it asks the concerned video server to improve it.

QoS degradation and feedback control loop

In order to improve the QoS, the system increases or decreases the number of transmitted frames of certain types. To this purpose, we based our action on the characteristics of the standard MPEG format (ISO/IEC 13818-2, 1994), that defines a mechanism to code frames at the time of the video compression. When a video sequence enters the system, it is compressed and coded according to following three types of frames: *Intra frames* or I-frames (reference frames), *Predicted frames* or P-frames (which allow to rebuild a frame using an I frame) and *Bidirectional frames* or B-frames (which use I frames and P frames to rebuild a sequence). So, I frames are the most critical frames. In order to reduce the eventual resource overload, our approach consists in removing some frames from a video sequence in a controlled manner. We use a feedback loop which allows to stabilize the system during the instability periods (Bouazizi et al., 2005).
The approach is based on two principles: observation and auto-adaptation. The observation principle consists in observing the results obtained by the system and checking if the current QoS observed is consistent with the QoS initially required, e.g. in VoD application, the system checks if the video sequences are presented to users without interruptions.
The auto-adaptation consists in adapting the results according to the QoS required by the clients, by adjusting some network and video parameters, e.g. the system increases or decreases the number of accepted frames. The feedback loop ensures then the system stability. We propose in the following paragraph a new architecture of the master server based on the feedback control in order to control the QoS provided to users in case of multimedia system overload.

How the master server works?

Master server architecture

In the master server architecture, the monitor allows to collect the remaining capacity of every Video Server (VS) and distinguishes between the following two types of customer demands:

- a new video.
- a QoS modification.

We are interested by the QoS modification requests, which are more critical. Indeed, new video demands will use more resources and will probably disrupt the video packets transmission, decreasing then the QoS provided to customers. According to the values of the remaining video servers capacities, the admission controller determines whether or not a demand will be accepted in the system. This control is done by comparing the capacity required by the task and the capacity provided by the video server. This value E is given by:

$$E = capacity(VSV)\text{-}capacity(demand)$$

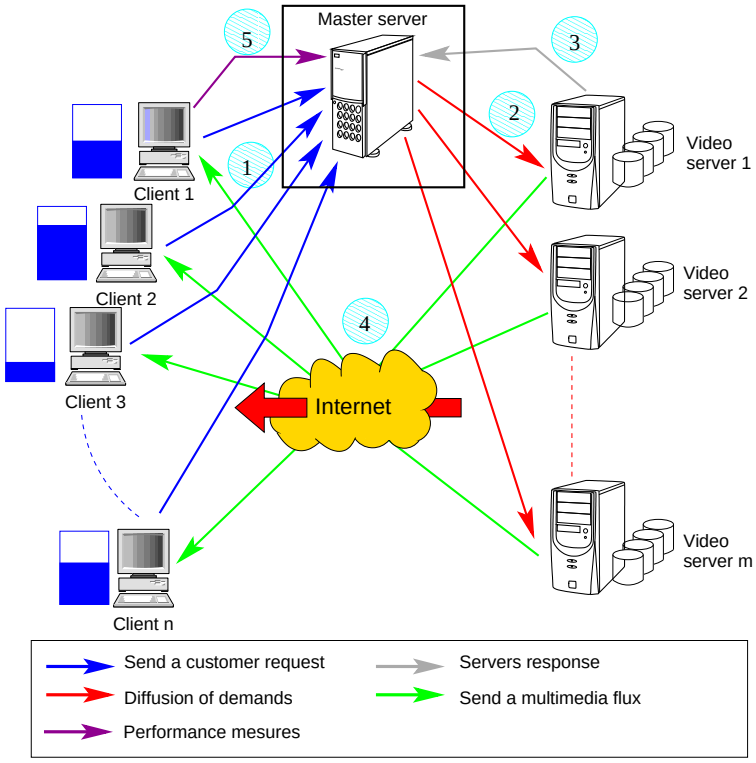


Figure 1: Functional model of the FCA-DMS architecture.

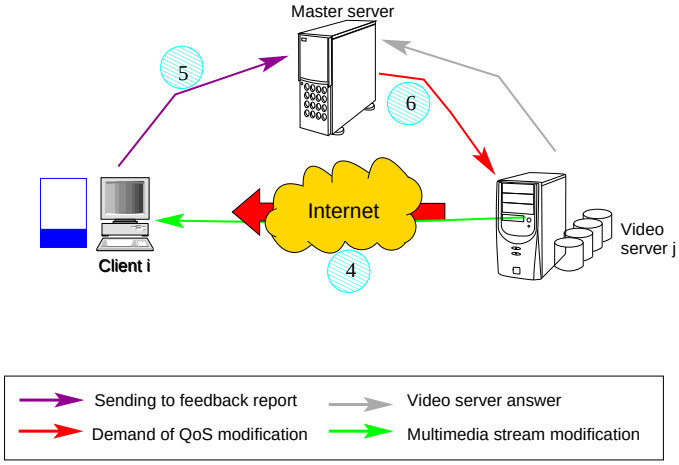


Figure 2: Adapted feedback loop for multimedia applications

Where:
capacity (VSV): capacity of video server.
capacity (demand): capacity needed for a new demand.
If E $\geq$ 0 then the system accepts the demand else
If E $<$ 0 the demand is aborted by the system or it is presented if it exist an other VS not saturated.
The QoS controller task takes the decision to increase or to decrease the QoS, according to the comparison of the current demand value with the last value measured.

- *Decreasing of QoS:* When the QoS controller decides to decrease the QoS, no problem occurs. It sends immediately a message to concerned VS and asks them to reduce the QoS. The advantages here are (1) to increase the server capacity on one hand, and (2) to free some resources that might be exploitable in case of QoS increase demand, in the other hand.

- *Increasing of QoS:* the video server in charge of a video may not find enough resources to answer to the customer demand. For example, if the VS ca-

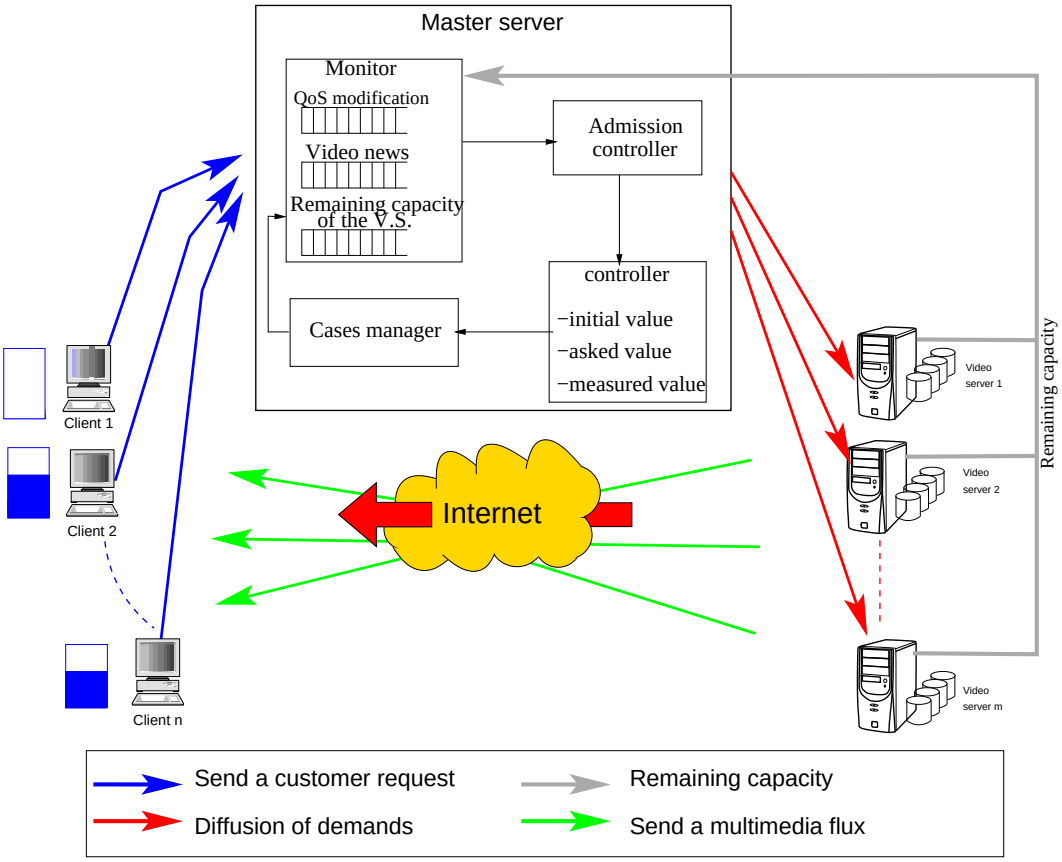


Figure 3: Master server architecture

pacity is 20 frames by second (f/s) and three customers ask a QoS of 15 f/sec of the same VS, the probability of the load of the system increases. The admission controller has autorized this demand to enter the system because it has checked that it exists one or several not saturated VS.

Replication strategy

A video server can distribute only videos stored on its disks. If a video is not accessible on several servers [Wolf and al. 95] (one VS contain this video), the probability that this last VS will be saturated increases. Therefore, it is necessary to define a new distribution strategy (*Algorithm 1*) of video packets in order to have another video server which is used to answer to the customer request. The saturated video server sends a request to the nearest video servers. Each video server behaves according to one of the following three scenarios:

1. It possesses the video and it is able to treat the request (it is not saturated).

2. It possesses the video but it is unable to treat the request (it is saturated).

3. It doesn't possess the video, but it is probably able to treat the request because it is not saturated.

In the two first cases, the replication strategy is not established. In the last case, the case manager, that has to control replication, sents an order to the saturated VS to start the replication. Consequently, the case manager elects a VS among those that answered and that are not saturated. The choice of the VS is done in order to get the best possible QoS. The demand returns again to the monitor, in order to allow to terminate the replication, then the monitor restarts works.

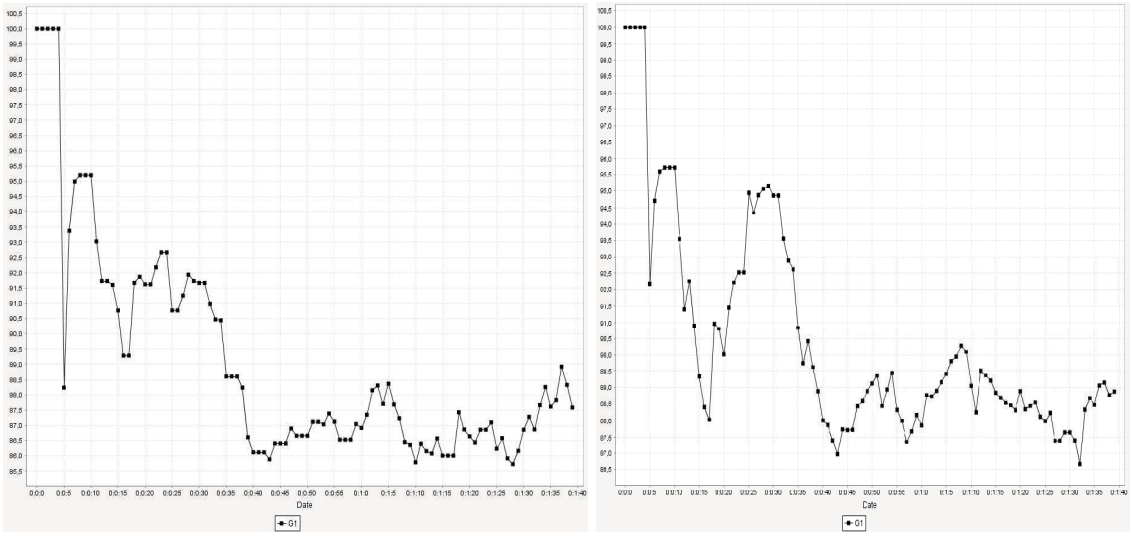
Simulations

Presentation of the simulator

To test the feasibility of our architecture, we had designed a simulator in order to verify the system behavior and its adaptation in relation to load variations. This simulator takes the architecture components presented in the Section . A master server is placed to video servers participating to the video diffusion. Thus, this service permits to organize the server addition while assigning them a number and referencing the accessible objects. After the master server start, the video are available in

Algorithm 1: Replication strategy

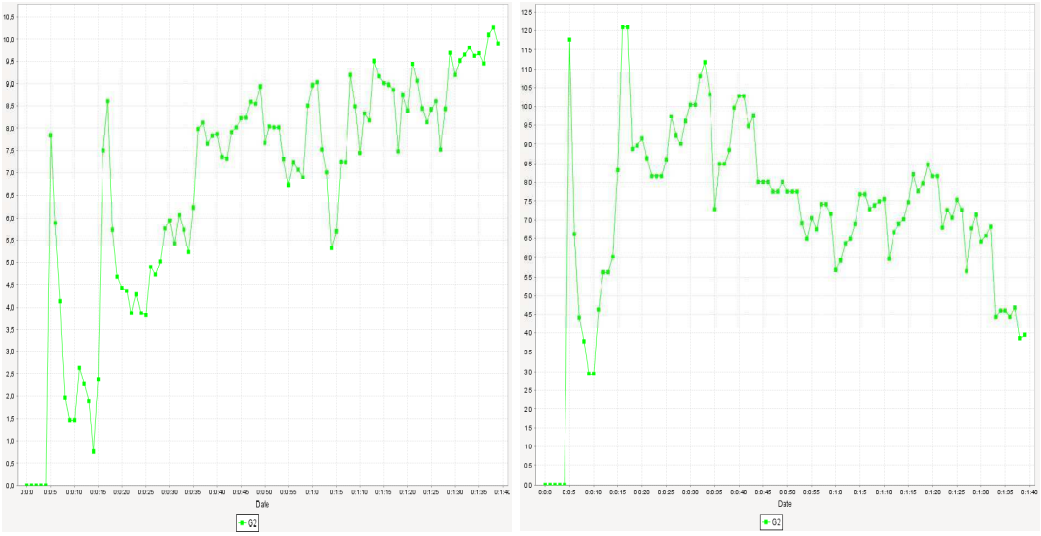
```
master server : MS
client : cl
video server : VS
neighbour VS : CVS
selected CVS : SCVS
begin
        SCVS = Ø
        VS is saturated and has the video
        For all CVSi do
                send-request (VS, CVSi)
                if (CVSi has not the video and not saturated) then
                        SCVS = Ø
                        exit-for
                else
                        if (CVSi has the video and is not saturated) then
                                put CVSi in SCVS
                        end if
                end if
        end for
        if (SCVS ≠ Ø) then
                choose (CVSj in SCVS)
                video-replication (VS, CVSj)
        end if

end
```



(a) Without replication strategy (b) With replication strategy

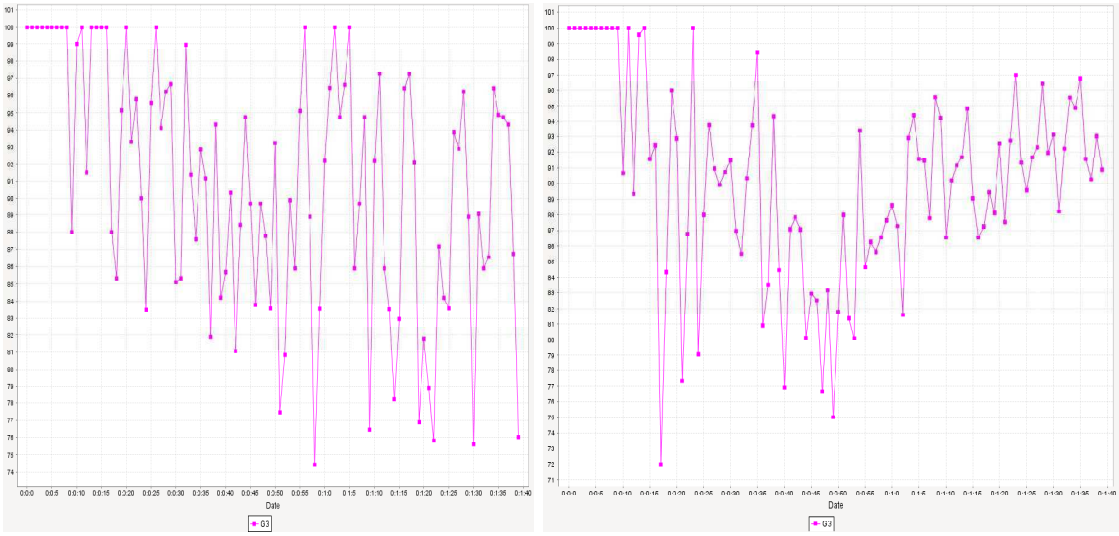
Figure 4: received frames rate without and with replication strategy.

the video distribution references at the master server and gets a number. To make a request, the client program have to ask to the master server, that distributes the request to available video servers. When the master server designated a video server able to answer to the client request - indeed that it arranges the asked content and it is able to provide the required QoS - it sends the video server reference to the client. Then, the streaming from the video server may begin. After some time of streaming, a return on the quality of service received by the client is sent to the master server. If it does not exist an available VS for the new video demand, the MS manages the QoS improved demand. It is based on the replication strategy in order to improve the



(a) Without replication strategy (b) With replication strategy

Figure 5: lost frames rate without and with replication strategy.



(a) Without replication strategy (b) With replication strategy

Figure 6: useful frames rate without and with replication strategy.

QoS requested by the client. This simulator has been implemented in JAVA language and a object modeling language to design the three parts of the architecture.

Simulation objectives and results

Simulations have as main goal to show, how our simulator allows to efficiently adapt the QoS of client according to the current system load. Notably, the system must adjust when the client number that effect requests varies in the time. System overload comes from different sources; for example, a large number of clients making requests in our system. We can limit this client number by a monitor localized at the master server level. Also, incontrollable use of the network such as, network resource used by other applications can cause the congestion. Our architecture adjust itself while reducing the QoS stored to the different client.
In figures 4, 5 and 6, we have three curved lines G1, G2 and G3 that indicate:

- *G1:* received frames rate.
- *G2:* lost frames rate.
- *G3:* useful frames rate.

In theses figures, we noticed a difference in lost frame rate and received frames rate to the customer and useful frames rate. In the figure 4(a), we observed lost frames rate, received frames rate and useful frames rate before using the replication strategy of video contents in case of system overload.

In the figure 4(b), we noticed the same rate after the implementation of replication strategy. After sometimes, we notice in the figure 4(b) that loss frames rate is decreasing also increasing the received frames rate and useful frames rate. We have not noticed an important gain for the received frames rate and the useful frames rates level, because our replication strategy is not sufficient to stabilize desire QoS by the client. But, by combining our replication strategy with other techniques such as bandwidth sharing also the storage technical of the VS, we can have very important gains in received frames terms. Our simulation shows almost 10% of gain of the loss frames rate. In the new MS architecture, frames loss become somewhat controlled, since, our new architecture is also based on observation and auto-adaptation technique and on the principle of demand selection.

Conclusion and futures works

While current resource management systems provide mechanisms which provide reliability with respect to QoS, it is not sufficient since there are many well established application scenarios where QoS management is required, e.g., distributed multimedia systems. Our main contribution is related to the adaptation of master server architecture and the establishment of a video replication strategy. A possible extension of this work is the enhancement of the architecture, that we have presented, in order to bring some fault tolerance because of the presence of only one master server.

We have also presented the importance of the master server architecture improvement and have given a certain priority to the QoS modification demand, in order to increase its reliability and robustness and to converge towards the QoS specified by the client.

Simulations results allow us to validate the feasibility of our approach and should allow to provide results demonstrating the real contribution of this new approach.

An other possible future work would consists in building a real video on demand server based on the architecture that we propose. We will take into account of frames storage management and frames organization in video servers, notably, the most efficient manner (from QoS point of view) to videos broadcast between the different video servers. This work requires to compare the performances obtained when using each manner to organize and store videos.

REFERENCES

Amirijoo M.; Hansson J.; and Son S.H., 2006. *Specification and Management of QoS in Real-Time Databases Supporting Imprecise Computations. IEEE Transactions on Computers*, 55, no. 3, 304–319.

Amirijoo M.; Hansson J.; and Song S., 2003. *Error-Driven QoS Imprecise Management in Imprecise Real-Time Databases.* In *Proc. of Euromicro Conf. on Real-Time Systems (ECRTS'03).* Portugal.

Bouazizi E.; Duvallet C.; and Sadeg B., 2005. *Management of QoS and Data Freshness in RTDBSs using Feedback Control Scheduling and Data Versions.* In *Proceedings of* 8^{th} *IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC'2005).* Seattle, United State.

Dulgheru N., 2004. *Management of QoS in Distributed MPEG Video.* Master's thesis, University of Linkoping.

ISO/IEC 13818-2, 1994. *Information Technology-Generic Coding of Moving Pictures and Associated Audio, Part 2: Video.* Recommendation ITU H.262, International Standardization Organization.

Kang K.; Son S.; Stankovic J.; and Abdelzaher T., 2002a. *A QoS-Sensitive Approach For Timeliness and Freshness Guarantees in Real-Time Databases.* In *Proceedings of the Euromicro Conference on Real-Time System.* 203–212.

Kang K.D.; Son S.; and Stankovic J., 2002b. *Service Differentiation in Real-Time Main Memory Databases.* In 5^{th} *IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (IEEE ISORC'02).* Washington D.C, 119–128.

Ng J.; Leung K.; and Wong W., 2000. *Quality of Service for MPEG Video in Human Perspective.* Tech. rep., Hong Kong Baptist University.

Ramamritham K., 1993. *Real-time databases. Journal of Distributed and Parallel Databases*, 1, no. 2, 199–226.

Ramamritham K.; Son S.; and DiPippo L., 2004. *Real-Time Databases and Data Services. Real-Time Systems*, 28, 179–215.

Wu L.; Faloutsos C.; Sycara K.; and Payne T., 2001. *Multimedia Queries by Example and Relevance Feedback. IEEE Data Engineering Bulletin*, 24, no. 3, 14–21.