Multimedia Tools and Applications, 23, 75–101, 2004

A Logical Approach to Quality of Service Specification in Video Databases

ELISA BERTINO bertino@dsi.unimi.it
Dipartimento di Scienze dell'Informazione, University of Milano, Via Comelico, 39/41 20135 Milano, Italy

AHMED K. ELMAGARMID ake@cs.purdue.edu
Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA

MOHAND-SAÏD HACID mshacid@bat710.univ-lyon1.fr
UFR d'Informatique, University Claude Bernard Lyon 1, 8, blvd Niels Bohr 69622 Villeurbanne, France

Abstract. Quality of Service (QoS) is defined as a set of perceivable attributes expressed in a user-friendly language with parameters that may be objective or subjective. Objective parameters are those related to a particular service and are measurable and verifiable. Subjective parameters are those based on the opinions of the end-users. We believe that quality of service should become an integral part of multimedia database systems and users should be able to query by requiring a quality of service from the system. The specification and enforcement of QoS presents an interesting challenge in multimedia systems development. A deal of effort has been done on QoS specification and control at the system and the network levels, but less work has been done at the application/user level. In this paper, we propose a language, in the style of constraint database languages, for formal specification of QoS constraints. The satisfaction by the system of the user quality requirements can be viewed as a constraint satisfaction problem, and the negotiation can be viewed as constraint optimization. We believe this paper represents a first step towards the development of a database framework for quality of service management in video databases. The contribution of this paper lies in providing a logical framework for specifying and enforcing quality of service in video databases. To our knowledge, this work is the first from a database perspective on quality of service management.

Keywords: Quality of Service, multimedia presentations, video databases, QoS parameters, QoS specification, QoS mapping, constraint-based query languages, constraint satisfaction, constraint optimization, reactive systems

1. Introduction

There is a qualitative difference between time-based media and the forms of data traditionally stored in database systems. Time-based media, including digital video and digital audio, music and animation, involves notions of data flow, timing, presentation, etc. These notions are foreign to conventional database management systems.

Since the usefulness of time-based presentations depends on the accuracy of both timing and data, the quality of the presentation is as important as the correctness of the query. Where database design has traditionally been concerned with the delivery of correct results with acceptable delay, multimedia systems present a new challenge: to deliver results with *acceptable quality* in real-time. But how accurate must be a presentation to be acceptable, and how can we guarantee that a presentation achieves that accuracy?

Typical application QoS parameters for images and video include image size, frame rate, startup delay, reliability, etc. The application QoS profile can also include subjective factors such as the degree of importance of the information to the user and the overall cost-quality metric that the user desires. Network QoS parameters include bandwidth, delay, jitter and loss rate. End-system parameters include CPU load, utilization, buffering mechanisms and storage related parameters. Users express dynamic preferences for media quality through benefit functions, e.g., (1) frame rate benefit function which indicates that beyond a threshold frame rate, there is no additional benefit, (2) synchronization benefit function which indicates that the benefit is high only if the audio/video synchronization skew is low.

One particular problem that has been proven to be challenging to solve involves the specification of quality of service. "The human user of a multimedia application is the starting point for overall QoS considerations". In the end, it is users of applications who are interested in the level of quality of service being delivered. Consequently, quality of service must be considered from the user's perspective, based on the user's expectations associated with applications. In other words, quality of service specifications must be application-level expectations, as opposed to low-level resource reservations.

User demands can be flexible. For example, some users accept only high quality video, while others are satisfied with lower quality when the system capacity cannot accommodate them otherwise. Some users allow service degradation as long as specified and agreed upon minimum quality is guaranteed. The system should be adaptable to accommodate various user's QoS requirements.

A deal of work has been done on QoS specification and handling at the system and the network levels, but less work has been done at the application/user level. This paper defines a methodology for QoS specification and enforcement. The definitions are intended to be general enough to apply to presentations in any multimedia system. We would like to be able to endow multimedia systems with capabilities that will make them able to decide whether a QoS specification is satisfiable or not. From the perspective of video database systems, the implementation of quality of service manager requires efficient algorithms for solving sets of constraints. A formal account of quality constraints is an essential step in demonstrating the correctness of such algorithms, and may yield more efficient processing strategies.

Paper outline. This paper is organized as follows. Section 2 summarizes the contributions of this work. Section 3 discusses related work. It gives a brief summary of some of the approaches proposed to tackle the problem of QoS in multimedia databases. Section 4 provides an introduction to QoS in time-based media systems. Section 5 gives an abstract architecture of a video management system incorporating a QoS manager. Section 6 describes the query language. The language allows to express search queries constrained by quality parameters. Section 7 develops an algorithm for deriving implicit constraints from explicit constraints that must be satisfied to guarantee the quality required by the user. Section 8 shows how the mapping between parameters of different layers in the system can be specified by simple rules. In Section 9 we provide a logical framework for negotiation. That is a calculus for computing best qualifiers for user-specified quality parameters. Section 10 describes the on-going implementation. We conclude in Section 11 by anticipating on the necessary extensions.

2. Contributions

There is currently considerable interest in developing multimedia applications. However, it has become clear that existing architectures for database management systems do not support the particular requirements of continuous media types such as digital video and audio. This is particularly the case in the important area of quality of service support for time-based applications.

Database technology offers many benefits for multimedia applications, such as high-level query languages, concurrency, and device and physical data independence. But current database systems do not adequately support time-based presentations. Relational data manipulation languages have demonstrated the value of letting the application specify *what* is wanted, and letting the database plan *how* to retrieve it. To support time-based presentations, a data manipulation language for a multimedia database should also allow the application to specify *when*, *where*, and *how precisely* the data should be delivered [23]. These constraints on delivery are an example of a QoS-based interface. For example, multimedia presentations are used extensively in many applications such as computer-aided training, computer-aided learning, world wide web sites, product demonstrations, document presentations, online books, electronic encyclopedias, etc. Currently, these multimedia presentations are created by using commercial multimedia authoring tools and stored into persistent storage such as a CD medium. Recently, commercial multimedia authoring tools have database access or a database front end to let users access media files and clip libraries. However, to the best of our knowledge, interaction between a multimedia authoring tool and a multimedia database is loose and the database is used for only very basic purposes. Additionally, these aspects are not investigated formally. We believe that multimedia presentations should be managed by multimedia databases and queried by an integrated query language to allow users to choose multimedia presentations with respect to their content.

None of the proposed data models for time-based multimedia that we are aware of supports queries for imprecise results. For example, Gibbs et al. [9] described a data model that captures the structure and synchronization relationships of complex time-based multimedia presentations. This model includes media descriptors that attach a quality factor, such as "VHS quality" or "CD quality", to each media object, but these labels describe the quality of the representation rather than the presentation. Without the notion of presentation quality in the data model, one would presume that all information would be preserved in the result of a query. In practice, information loss in a time-based presentation is inevitable and unconstrained by current data models.

We partition the QoS parameters into two subsets, namely application-dependent parameters and application-independent parameters. For example, most electronic commerce applications require multimedia presentation to the customer from the venders, and then the quality of audio and video is important in addition to images, text and numbers. Application parameters describe requirements for application services and are specified in terms of media quality and media relations. Media quality includes source/destination characteristics such as media data unit rate and transmission characteristics such as response time. Media relations specify relationships among media, such as media conversion and inter-stream synchronization. Researchers have yet to determine the best set of QoS parameters

Table 1. Examples of possible Quality of Service parameters.

Type	QoS parameter
Application-dependent	Frame Width
	Frame Height
	Color Resolution
	Time Guarantee
	Space Guarantee
	Resource Requirements
	...
Application-independent	Delay
	Jitter
	Reliability
	Throughput
	Bandwidth
	Packet Loss
	Speed of the Network
	Network Topology
	...

for multimedia systems. Table 1 shows some common QoS parameters in the multimedia community (mainly for video). Relevant quantitative measures of system parameters such as CPU load and Network utilization for various user QoS specifications can be found, e.g., in [1, 5, 22, 25, 30, 35].

This paper advocates the use of constraint-based rule language for specifying and reasoning on application-dependent quality of service parameters in video databases. The framework presented here integrates techniques developed in constraint databases and Constraint Logic Programming (CLP). The main contributions of the paper are the following:

- We propose a query language, based on a CLP-scheme, for video databases which accommodates QoS parameters (mainly presentation parameters). As in [28], we consider queries to be composed of two parts: a *content* part and a *view* part. The *content* part specifies conditions that video sequences should satisfy to be answers to the query. The *view* part specifies constraints for a desired presentation of the outputs.
- We present a terminating procedure, called elaboration, allowing to derive implicit constraints from explicit ones stated by the user. The complete set of constraints will be used to build a presentation schedule and a retrieval schedule.
- We show how constrained rules can be used to map parameters of different layers in the system.
- When no retrieval schedule can be found to satisfy a presentation schedule, then some quality parameters have to be relaxed. We show that a framework based on preference logic programming constitutes a nice basis to achieve the goal of what parameters to relax in case of many choices.

There are several advantages to using a rule scheme, among them: (1) each rule represents a small, independent piece of knowledge—this facilitates modularity, (2) rigid syntax affords the convenience of checking consistency, and (3) it is easy to furnish explanation facilities.

Our formulation of quality of service and the problem of its satisfaction by a query offers the benefits of having a simple declarative semantics, providing modularity, and being amenable to an efficient implementation. Many of quality of service aspects have been considered in previous work, and one of our goals is also to unify ideas, provide a more formal foundation, and express these aspects in a way suitable for reasoning. To the best of our knowledge, this is the first logical framework combining techniques from constraint databases and constraint logic programming for specifying and handling quality of service in video databases.

Although in the basic form that we give here, the formalism does not account for all aspects of quality of service in video databases, it constitutes a kernel to be extended. Showing how we can formally specify and reason about quality of service is useful and significant. We hope that this work opens up a number of possible future research on incorporating quality of service management in multimedia database systems.

3. Related work

Our work relates to several fields of research regarding support of quality of service in multimedia databases. We briefly discuss the relationship to time-based media (mainly video) presentation, quality of service mapping and specification in multimedia databases. This section is intended to be illustrative. We apologize if we left out other relevant references.

- *Time-based media presentation.* In [36], the authors proposed an extension of existing object-oriented database techniques to include mechanisms for video presentation. Compressed video data streams are segmented and stored as sets of video objects coupled with specified synchronization requirements. The only quality problem considered in that paper is the one concerning synchronization between the decoder and the viewer of frames. This is done by means of buffer management. User/application QoS is not addressed. Lee et al. [20] proposed a graph data model for the specification of multimedia presentations. Each node of a presentation graph represents a media stream. Edges depict sequential or concurrent playout of streams during the presentation. The paper assumes that multimedia presentations are created and stored in the form of multimedia presentation graphs, which can be viewed as high level abstractions for multimedia presentations. The emphasize in that paper is on finding appropriate language constructs to create and manipulate presentation graphs rather than constraining presentation during database querying. Operators like *Next*, *Connected*, and *Until* and path formulas are used for querying presentations. Hence, augmenting a database by a presentation base could help users to efficiently locate the desired information in the database. Note that with this approach, the quality parameters attached to graphs are statically checked. In contrast, we allow a dynamic specification and checking of quality parameters. Another problem with this approach is that it is not clear which presentations to store in the case of very large databases and how retrieval of presentations can be combined with data retrieval. Wu et al. [34] examined

the querying requirements of large libraries of multimedia presentations. The authors proposed an integrated composition and query capability to permit the reuse of multimedia objects, presentations and presentation segments. Here the quality of service is seen as the benefit gained in retrieving and reusing components of presentations. Again with this approach users cannot dynamically specify quality of service parameters as they are already integrated and compiled in pre-defined presentations. Staehli et al. [28] defined a methodology for QoS specification regarding presentation. In the proposed framework, a user may control a player's view parameters such as window size and playback rate, as well as quality parameters such as spatial and temporal resolution. When a user chooses to begin a presentation, the player needs to verify that a presentation plan consisting of real-time tasks will satisfy the QoS specification. The QoS constraints are specified using the *Z* specification language [27] which is not intended to be manipulated by end-users. In the framework we propose the constraint part of our queries can be translated into the specifications of [28] and then their methodology can be applied to the specifications .

- *QoS Mapping.* In order to guarantee the fulfillment of user/application requirements, a mapping onto the involved operating system and network resources has to be performed. QoS mapping is regarded as the process of translating QoS parameter bounds from layer to layer, and finally, to resources, e.g., buffers. Kim and Nahrstedt [17] presented an integrated view of translation and admission control relations for MPEG-video streams between a multimedia distributed application such as video-on-demand, and its underlying system resource, CPU. Knoche and de Meer [18] proposed guidelines as to how QoS parameters are affected by stimuli. Fischer and Keller [8] presented a set of translations of QoS parameters from the application's point of view into transport and operating system QoS parameters. The mapping is a strict one in the sense that no framework is given for negotiation.

  These proposals consider mapping as an internal operation carried out by the systems. It is not clear how the operational level of this mapping is specified. In our framework, we specify the mapping by means of declarative rules. Rules conditions specify the input (source) quality parameters and rules conclusions specify the target parameters. These target parameters are computed by executing the set of rules on a given quality of service specification expressed by the user.
- *Specification of QoS.* Lakas et al. [19] introduced an approach to the formal design and modeling of QoS parameters in multimedia systems. The proposed approach is based on the principle of separation of concerns. The authors use the process algebra based language LOTOS [2] to specify the functional behavior of the system and separately describe the quality of service requirements using an appropriate temporal logic. It is not clear in this paper how the mapping is done and what should be the operational semantics that will be used to perform the satisfiability test of the QoS constraints. Gibbs et al. [9] proposed a conceptual data model for time-based media. The proposed data model includes media descriptors that attach a quality factor to each media object. These media describe the quality of the representation rather than the presentation. This kind of data can be useful for (semantic) query optimization in the case where the user makes reference to these metadata when specifying queries. This information can be taken into account in our framework as constraints in the view part of queries and it is

handled prior to the evaluation of the query (provided that sufficient local information is available). Walpole et al. [31] proposed a model for the specification of user-level QoS preferences in multimedia databases. The quality is defined as a distance measure in multiple quality dimensions, and utility functions are used to capture user QoS preferences in each dimension.

4. Quality of Service in time-based media systems

Recent advances in computer and network technology have made the development of multimedia systems a topic of main importance, with the farther target to make them the ultimate systems for providing high quality complex services in many aspects of the human life. In particular, distributed multimedia systems become more and more complex, in terms of the variety of services they provide and the management of the underlying physical infrastructure that implements these services. Beyond the inherent complexity of such systems, the situation is further complicated if user requirements are to be considered.

As the user is the starting and the ending point in such systems as multimedia servers, QoS expresses the *proximity* between the system's response and user's demands. In general, QoS comprises a set of technical and other parameters of the system that control its functionality and that must be tuned based on user satisfaction. Quality of Service management in the context of multimedia systems, sets the appropriate parameters and reserves the necessary resources, so as to achieve the required functionality and optimize the performance of the overall system.

In this paper, quality of service is the overall quality, as perceived by the user of a service. A QoS context is a set of interrelated parameters affecting the quality of service. Even though there might be a generic relationship between any two QoS contexts, finding such a relationship is practically very difficult. Therefore a mapping functionality between two specific QoS contexts is required to present this relationship.

Multimedia database systems are being extended to support presentations of continuous media, such as video and audio, as well as synthetic compositions such as slide shows and computer generated music. We call these presentations time-based because they communicate a part of their information content through presentation timing. While applications with text and numeric data types expect correct results from database queries, the real-time constraints of time-based presentations commonly make it impossible to return complete and correct results. Some information loss is also unavoidable in any conversion of continuous media between analog and digital representations.

The most common multimedia presentation tools use a best-effort approach, which aggressively consumes resources to present all data as promptly as possible. When resources are overloaded, a best-effort presentation will lose information. Many researchers have demonstrated best-effort systems that maintain approximate synchronization despite variable latencies and low resource availability. Those systems show that a presentation can be acceptable even when quality degradation is noticeable. However, the best-effort approach has two problems: First, if perfect presentation is not necessary, why should a multimedia system expend extra "effort" to achieve the best quality? Second, how much quality degradation can be allowed when many real-time presentations compete for scarce resources? If

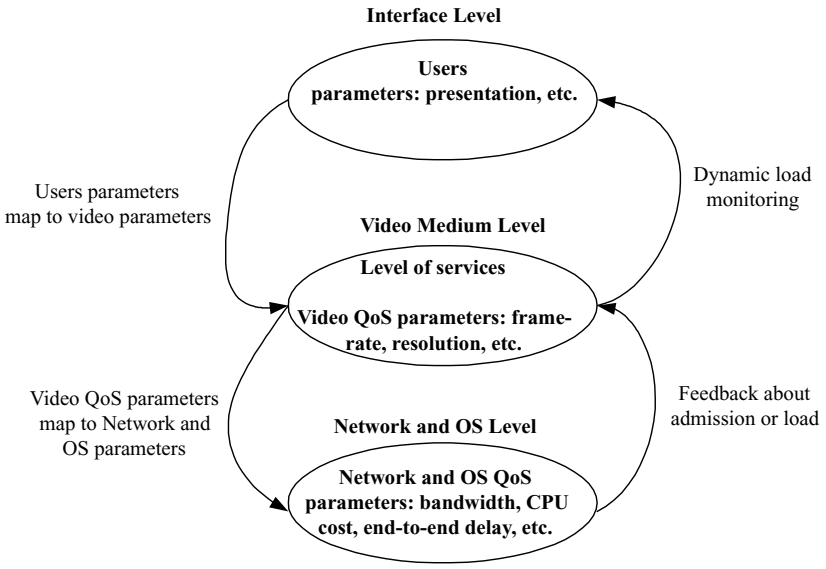


Figure 1. Relationships between the system's layers regarding QoS.

any application is to be guaranteed acceptable service, some information is needed about presentation QoS requirements. Other researchers have recognized that best-quality presentations are often too expensive and unnecessary. The Capacity-based-Session-Reservation-Protocol, for example, supports reservation of processor bandwidth from the specification of a range of acceptable spatial and temporal resolutions for video playback requests.

Awareness driven video QoS involves establishing a mapping between the high-level, medium-independent spatial model mechanisms and low-level video specific QoS parameters. This is shown in figure 1. The top-most level of the mapping is the interface level. The video medium level is concerned with providing both generic and application specific video services. For example, a conferencing tool might define two levels of service, a high resolution service to be associated with the most active or interesting speakers and a low resolution service to be associated with passive or more peripherally interesting speakers. The lowest level is the network and operating system level. This level is concerned with general (i.e., video independent) low-level QoS parameters such as bandwidth, error-rate, end-to-end delay and CPU time.

5. A system architecture supporting Quality of Service

This section proposes an abstract architecture for a video data management system. This abstract architecture, inspired from the one given in [13], is designed to provide a management system for video data. Figure 2 shows the block level architecture of a video data management system integrating quality of service management. The function of each of

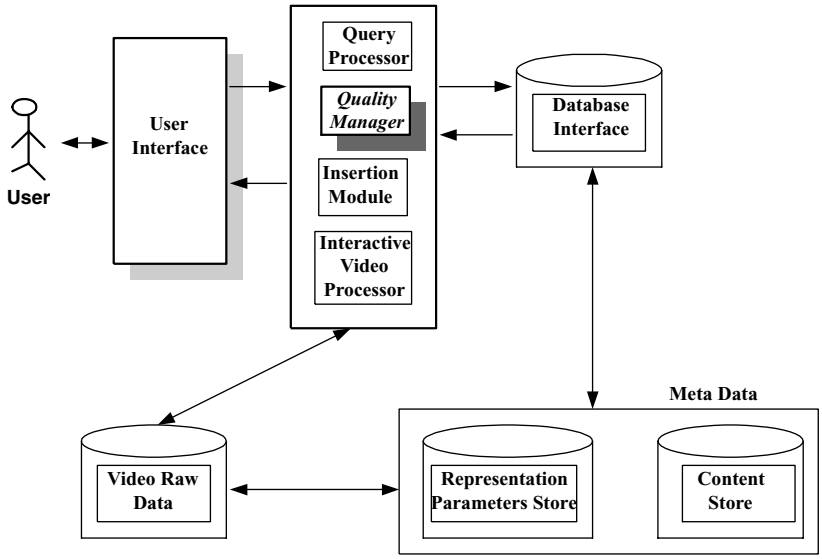


Figure 2. Video database management system.

these blocks is described below.

User Interface. The user interface in a video data management system plays a critical role in the overall usability of the system. It allows the specification of queries to the database.

Query Processor. The basic function of the query processor is to generate queries execution plans. It performs the transformation between the queries formulated by the user into a data model representation which can be used to locate the data. The user queries can be specified in a variety of different languages. In this paper we consider queries formulated in a constraint-based query language. This will be further detailed in Section 6.

Quality Manager. This is the main module we are interested in. This module deals with the enforcement of the quality of service specifications. As we will see, quality of service is specified by means of a set of *quality constraints*, and quality constraint checking is performed by this module based on the algorithm that we will provide in Section 7.

Insertion Module. This module deals with the raw video data as it is being inserted into the video database.

Interactive Video Processor. It is used to browse and play-back retrieved video sequences. A user may control view parameters, such as window size and playback rate, as well as quality parameters such as spatial and temporal resolution. In this case, the quality manager module is expected to guarantee the *satisfaction of quality constraints*.

Database Interface. This module provides the interface between the video processing and video query software, and a database management system. The interface translates the video queries formulated by the user into appropriate queries which constitute inputs to the Query Processor.

Representation Parameters Store. This module is the database that stores the video representation parameters. Each unit of video in the database is represented by an instance of the video data model. The exact nature of the data model depends on the type of application and the nature of the queries supported by the video data management system. Examples of parameters are color model, duration, etc.

Content Store. This module is the database that stores data referring to content semantics. This data is also referred to as content-descriptive metadata. It is concerned with relationships of video entities with real-world entities or temporal events, emotions and meaning associated with visual signs and scenes.

Video Raw Data. This is the physical store for the video data. The video data may be in a compressed digital form.

6. Expressing queries involving QoS/QoP[1]

Definition 1 (Predicate symbol). We define the following predicate symbols:

- each $\mathsf{P} \in \mathcal{R}$ with arity n is associated with a predicate symbol P of arity n,
- a special unary predicate symbol video. It can be seen as the class of all video sequences. Each video sequence is seen as an object.
- a special unary predicate symbol Object. It can be seen as the class of all objects other than video sequences.

So, the only two extensional relations, besides those explicitly stated (i.e., $\mathcal{R}$), are video and Object. Additionally, we have pre-defined predicates for defining spatio-temporal relationships of a presentation.

We also assume the existence of the unary predicate *display_abstract*. *display_abstract* will be used when the user requires only a browsing version of the frame (i.e., only representative frames are returned). By default complete video sequences are returned. This means that for each video sequence, the system maintains a short version (i.e., representative frames) and a full version. The predicate *display_abstract* is not taken into account during query evaluation, but interpreted by raw video retrieval component.

A query is composed of two parts: *content* part and *view* part. A *content* specification defines a set of logical video sequences to be retrieved in the database. *A view* specification maps content onto a set of physical display regions or parameters by specifying desired constraints. Quality is then a measure of how well an actual presentation of content specification matches the ideal presentation of content on a view.

By allowing independent control of content, view and quality, a video system can offer a wider range of services that take advantage of the flexibility of computer platforms. As an example, consider the presentation of video. After selecting the content for presentation, a user should be able to choose view parameters and quality levels. For example, the user may choose a view with 640×480 pixel display window, but a quality specification that requires only 320×240 pixels of resolution. In this case the player may be able to avoid generating the full resolution images from a video sequence.

Definition 2 (Query). A query is of the form:

$$Q \parallel S$$

where Q is the content part of the query, that is the characterization of the set of videos that will be retrieved from the database. S is the quality part of the query, specifying a set of constraints that retrieved video sequences should satisfy in order to be displayed. Each video sequence satisfying the *content* part Q will be displayed by maintaining the quality of service specified in the *view* part S.

Example. The following query

$$\begin{aligned} \text{ans(V)} \leftarrow\ & \text{video(V), category(V, ``movie''),} \\ & \text{produced\_by(V, ``Steven Spielberg''),} \\ & \text{date(V, ``1990'')}\parallel \\ & \text{display(V, W), coord\_x(W, X),} \\ & \text{coord\_y(W, Y), resolution\_x(V, } V_x\text{),} \\ & \text{resolution\_y(V, } V_y\text{), } X \leq 250, \\ & Y \leq 200,\ V_x = 320,\ V_y = 240 \end{aligned}$$

retrieves video sequences of type "movie" produced by Steven Spielberg in 1990. Each video sequence, in the answer to this query, will be displayed on the screen in a window 250×200 with a resolution of 320×240.

Predicates appearing in the *view* part of a query may be divided into "local" quality predicates and "external" quality predicates. "external" quality predicates are those involving data of the video representation parameters store (see figure 2).

Example. In the following query

$$\begin{aligned} \text{ans(V)} \leftarrow\ & \text{video(V)}\parallel \\ & \text{quality\_factor(V, ``VHS quality''),} \\ & \text{color\_model(V, ``RGB'')} \end{aligned}$$

quality_factor and *color_model* are two external quality predicates.

Intuitively, a medium (such as a video server) may have data stored on it in a given format (e.g., raster, bitmap, vhs format, pal, secam, etc.), in some storage (e.g., video tape), together with some functions (e.g., display from frame n to frame m).

7. Constraints derivation for QoS enforcement

When creating a multimedia presentation, three basic questions must be answered:

- What objects should be included in the presentation?
- When should these objects be presented to the user?
- Where should the objects appear on the screen?

These three questions can only be answered by the individual who creates the presentation (called the author of the presentation). Once the above questions have been answered by the author, a presentation schedule can be created that will specify *when*, *where* and *how* someone viewing the presentation will actually see the objects constituting the presentation.

Informally speaking, the answers to the when, where and how questions are most naturally expressed through the use of *constraints*. Different solutions to the when and how constraints yield different presentation schedules. Clearly, the choice of a constraint language within which such constraints are expressed plays a central role in the types of temporal/spatial relationships that can be expressed, and the efficiency with which such constraints can be solved.

Once a presentation schedule has been created, we need to create a retrieval schedule that ensures that the resources needed to deliver the presentation to the client are in fact available. Such resources may include availability (load and buffer) of remote data servers, availability of bandwidth from the network, and the availability of buffer space at the client.

Figure 3 shows the cycle of how presentation schedules and retrieval schedules interact. The user specifies a retrieval query augmented with quality constraints. The augmented query and the metadata are fed to a module called evaluation and elaboration, which retrieves answers to the query and derive additional constraints. For each constrained object retrieved from the metadata, the constraint solver solves the attached constraints. A solution is a presentation schedule. Any solution may be picked nondeterministically with a view to creating retrieval schedule for it. If this is possible, then we don't need to go further. If no retrieval schedule can be created for a specified presentation schedule, then we must pick another presentation schedule. This cycle is continued till a presentation schedule is found that has a corresponding retrieval schedule.

7.1. Evaluation and elaboration

Our idea is to use information from the original query to constrain the search for the objects, and to use intermediate tests to eliminate useless partial solution tuples as soon as possible.

Given a query $Q \| S$, the first step consists in evaluating Q to find the set of objects answers to the query. For each retrieved object O we build two sets $\mathcal{Q}.\mathcal{S}$ where $\mathcal{Q}$ is called the facts set and $\mathcal{S}$ is called the constraints set. $\mathcal{S}$ is the union of S and the constraints attached to O in the metadata.

An *elaboration rule* is an if-then rule that adds constraints to a set of constraints.

Elaboration is applied to a query for the purpose of making implicit constraints explicit, and then accessible to a negotiation algorithm. The propagator applies forward chaining rules to augment $\mathcal{S}$ with constraints that logically follow from $\mathcal{S}$ and $\mathcal{Q}$.

As an example, consider the following query:

video(X), audio(Y), X contains Z,
Z name "*Clinton*", X has_audio $Y \|$
display_time(t_d), $t_d \leq 5$, resolution(X, "*high*")

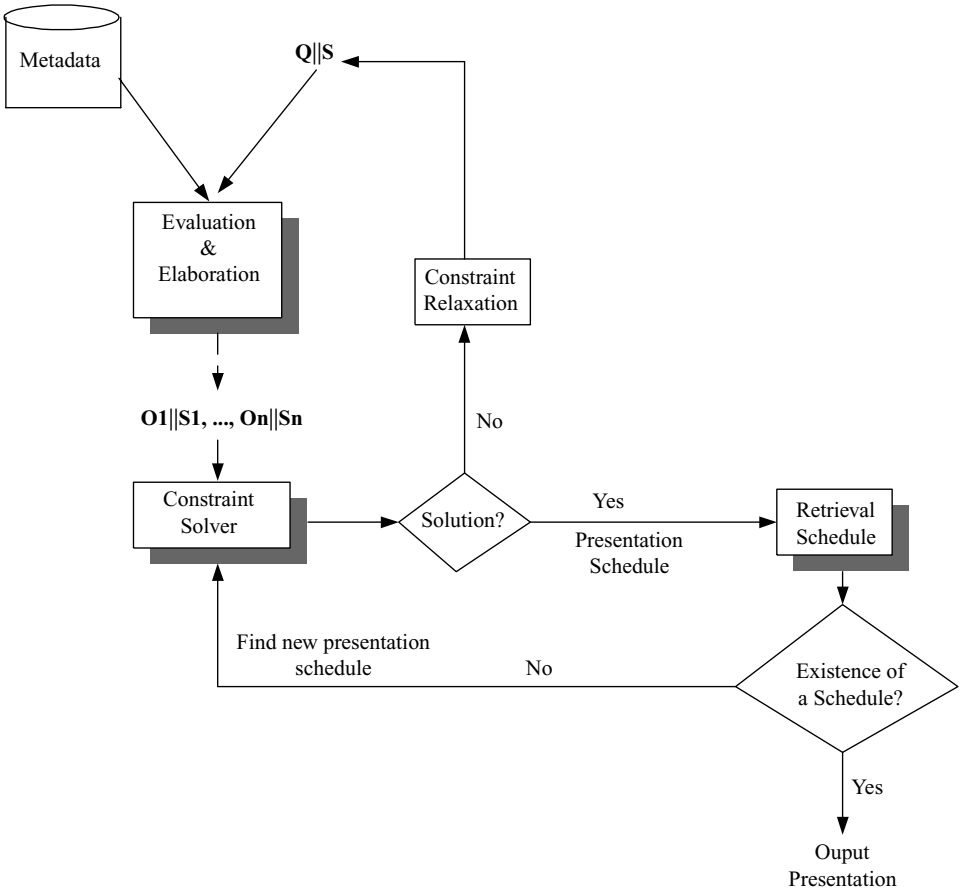


Figure 3. Interaction between the different modules.

Suppose the evaluation of this query returns the answer $\{X/v, Y/a, Z/o\}$. We have:

$$\mathcal{Q} = \{\text{video}(v), \text{audio}(a), v \text{ contains } o, o \text{ name } \text{“}Clinton\text{”}, v \text{ has\_audio } a\}$$
$$\mathcal{S} = \{\text{display\_time}(t_d), t_d \leq 5, \text{resolution}(v, \text{“}high\text{”})\}$$

By using the elaboration rule:

$$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{synchronize(X, Y)\}$$
if $\mathcal{Q}$ contains video (X), audio(Y), X has_audio Y

we can derive the atomic constraint

$$\text{synchronize}(v, a)$$

Now, by using the elaboration rule:

$$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \left\{t_{X_{\text{start}}} = t_{Y_{\text{start}}}, t_{X_{\text{end}}} = t_{Y_{\text{end}}}\right\}$$
if $\mathcal{S}$ contains synchronize(X, Y)
and $t_{X_{\text{start}}}, t_{Y_{\text{start}}}$ are variables denoting starting times of X and Y, respectively
and $t_{X_{\text{end}}}, t_{Y_{\text{end}}}$ are variables denoting ending times of X and Y, respectively.

we can derive the atomic constraints $t_{v_{\text{start}}} = t_{a_{\text{start}}}$ and $t_{v_{\text{end}}} = t_{a_{\text{end}}}$.

Let t_v be the display time that the video media can provide from $t_{v_{\text{start}}}$ and t_a be the display time the audio media can provide from $t_{a_{\text{start}}}$. Then, the constraints $t_{v_{\text{start}}} + t_d \leq t_{v_{\text{start}}} + t_v$ and $t_{a_{\text{start}}} + t_d \leq t_{a_{\text{start}}} + t_a$ are derivable by using the elaboration rule:

$$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{t_X + t_d \leq t_X + t_v, t_Y + t_d \leq t_Y + t_a\}$$
if $\mathcal{S}$ contains synchronize(X, Y)
and t_v is the display time the video medium can provide from t_X
and t_a is the display time the video medium can provide from t_Y
t_X and t_Y being the the starting time for display of X and Y, respectively.

The augmented set of constraints is then:

$$\begin{aligned}\mathcal{S} = \{&\mathsf{display\_time}(t_d), t_d \leq 5, \mathsf{resolution}(v, \text{`}high\text{'}),\\ &\mathsf{synchronize}(v, a), t_{v_{\text{start}}} = t_{a_{\text{start}}},\\ &t_{v_{\text{end}}} = t_{a_{\text{end}}}, t_{v_{\text{start}}} + t_d \leq t_{v_{\text{start}}} + t_v,\\ &t_{a_{\text{start}}} + t_d \leq t_{a_{\text{start}}} + t_a\}\end{aligned}$$

The final set of constraints (called completed set) is the one to which no elaboration rule applies. From the complete set, one derives presentation and retrieval schedules. The problem of scheduling a set of tasks with time and resource constraints is known to be NP-complete [21]. Effective heuristic algorithms exist for this problem [37] which are sensitive to the uncertainty in task completion times.

8. Interlayer constraints

The management, retrieval and display of video data require an environment where several systems cooperate. We consider such an environment as based on a location-independent object model where all interacting entities are treated uniformly as encapsulated objects. Objects are accessed through interfaces which define named predicates, implemented as internal operations. Activity takes place in the model when objects invoke named predicates in the interfaces of other objects. Hence in our architecture (see figure 2) each module is considered as an object encapsulating a set of internal operations. Additionally, we have two other important objects, namely *Network* and *OS* (*Operating System*).

So far, the quality manager dealt only with local constraints, that is presentation constraints or representation constraints, but not constraints (e.g., Jitter, Buffer size) regarding

the behaviour or the state of required resources for satisfying quality of service specification. Our extension to low-level layers is based on the notion of Universal Attachment. Universal Attachment [24] is a domain-independent mechanism for integrating diverse representation and reasoning methods into hybrid frameworks that contain a subsystem based on deduction over logical formulas. Predicate evaluation allows a deductive system to exploit external evaluation procedures by "attaching" programs to predicate and function symbols in a first-order language. When a symbol with an attachment is encountered during deduction, the attached program is invoked to calculate the appropriate value. As an example, we could attach the function symbol plus for a first-order language to the LISP function +. When the term plus(3,4) is encountered during the deduction process, the LISP program (+ 3 4) would be executed to evaluate it directly. Predicate evaluation has come to be referred to as procedural attachment.

Example. The following quality constraint

```
panic :-    video(M),
            frame-rate(M, F),
            F > 30,
            os-buffer-size(Z),
            Z < 6600
```

says that if one wants to display a video sequence with a frame-rate greater than 30 then one needs a buffer size greater than 6600 from the operating system. Here, os-buffer-size is a symbol with an attachment. The attached program will be executed at the OS level considered as an object encapsulating a set of operations and services.

8.1. Using rules to map Quality of Service parameters (or negotiation)

The high-performance characteristics of the networks (e.g., FDDI, ATM) enable new multimedia applications for which the standardized protocols and services no longer suffice. Especially in the area of service quality, the new applications need mechanisms to express and communicate their needs. It is quite important to provide mechanisms for QoS management, and in particular for mapping application-level parameters to communication and operating system QoS parameters.

The task of the network layer (with respect to QoS) is to provide the means to implement the service *semantics* defined for the transport service. The most important approach is the reservation of resources in the network (see, for example, [7]). Algorithms have been developed to limit delay and jitter, and to guarantee a certain throughput on internetwork connections. The parameters of this service are nearly the same as those for the transport service.

Achieving a certain QoS, especially in the software implemented layers, requires not only the support of the respective lower layers but also the operating system support. The main points here are CPU scheduling, memory management for buffering, and efficient file storage on mass media. Real-time CPU scheduling is extensively discussed in the literature (see, for example, [26, 29]). The aim of buffer management is to avoid jitter and to minimize

Table 2. Some user/application Quality of Service parameters.

Parameter	Domain	Qualifiers
period	Milliseconds	very fast, fast, normal, slow, very slow
quality	Integer	very high, high, medium, low, very low
reliability	Percent	very high, high, medium, low, very low
delay	Milliseconds	minimal, default
start offset	Milliseconds	minimal, default

Table 3. Examples of Transport/Network QoS parameters.

Parameter	Domain
Throughput	bytes per second
MTU (Maximum Transfer Unit)	bytes
Reliability	percent
Burstiness	integer
Delay	milliseconds
Jitter	milliseconds

copy operations [29]. This may be achieved by a large number of buffers which, however, may in turn lead to a waste of resources.

From the user's point of view, a large set of parameters is unacceptable and normally useless. Users do not want to specify numerous parameters which are often meaningless to them, such as jitter or cell loss ratio. In addition, they have no need to give exact values for certain parameters. A frame rate of 16 frames per second (fps) will look quite similar to a frame rate of 14 fps. Therefore only a small set of meaningful parameters should be offered to the user. As we said in the introduction, one of the most important set of parameters from a user's point of view are *presentation* parameters. Examples of such parameters, together with their possible qualifiers are given in Table 2.

Consider the set of network parameters given in Table 3. A possible mapping of the user's quality of service parameters of Table 2 to network parameters of Table 3 is depicted in figure 4.

This mapping can be specified by using our rules. Each mapping rule has the form:

$$D\_P : -S\_P$$

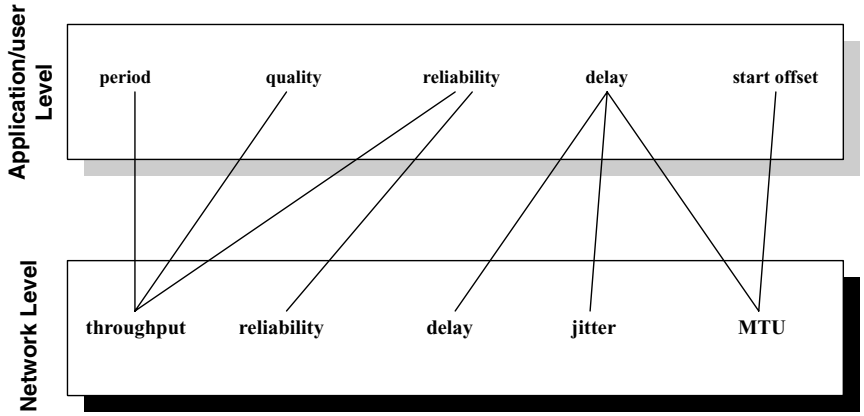


Figure 4. Mapping of application QoS parameters to network QoS parameters.

where S_P is the body of the rule which stands for the input parameters at the application level. D_P is the head of the rule specifying the desired target parameter for a given source parameter. The set of mapping rules should be defined on the basis of consultations with application designers and literature studies. Let us illustrate the dependency between *reliability* and *throughput* [8]. The reliability parameter controls the forward error correction scheme AdFEC (Adaptable Forward Error Correction). AdFEC adds redundancy to the stream to be transmitted such that lost parts of the original stream can be reconstructed. The percentage of parts that are retransmitted is dependent of the qualifier associated with the input parameter *reliability*. For example, if the user asks for a *low* reliability, then we have to use the AdFEC type *FEC_2_1* with a redundancy equal to 33%. This can be captured by the following simple rule:

$$\text{throughput}(\text{redundancy} \rightarrow 33, \text{AdFEC\_Type} \rightarrow \text{`FEC\_2\_1'})$$
$$\text{:- reliability}(\text{qualifier} \rightarrow X), X = \text{`low'}$$

Note that the syntax of this rule is a slight modification of the one used so far. This is in order to make things more explicit by making use of explicit label names (e.g., qualifier).

9. Negotiation

A considerable research work has addressed the issue of using resource management techniques to prevent fluctuations, i.e., performance variations. Typically, prior to the evaluation of a query, the QoS parameters are mapped into System and Network parameters. In general, this mapping process must result in a set of system resources allocated to the user's demand. A QoS negotiation process and an underlying QoS management infrastructure assure that all system components can provide their share of the requested resources during the session's lifetime. It was shown that this method works reasonably well in homogeneous

environments where resources allocation is straightforward and application sessions are not too complex. In many situations, however, this approach is too simple. Multimedia systems may be unable to sustain negotiated levels of QoS. Mechanisms should be developed in a way that allows systems to sustain resource fluctuations within certain limitations. This may be achieved by mechanisms that gracefully adapt to unexpected resource fluctuations.

In what follows, we provide an abstract, logical framework for dealing with resource fluctuations. We call the process negotiation. We show how a framework developed in Logic Programming with Preferences and Constraints for performing optimization and relaxation in a logically principled manner can be used to perform negotiation.

Optimization and relaxation (see, for example, [3, 12, 33]) are two important operations that naturally arise in many applications involving constraints, e.g., engineering design, scheduling, decision support, etc. In optimization, we are interested in finding the optimal (i.e., best) solutions to a set of constraints with respect to an objective function. In querying video databases, an optimal solution, with respect to the specified quality of service, may be difficult or impossible to obtain, and hence we may be interested in finding suboptimal solutions, by either relaxing the constraints or relaxing the objective function. In the following, we consider relaxation and optimization for negotiation.

First, we introduce some notions and results from Logic Programming with Preferences and Constraints that are necessary for our application (see, e.g., [11] for further details).

9.1. The PLP framework

A Preference Logic Program (PLP) may be thought of as containing two parts: a **first-order theory** and an **arbiter**. The first-order theory consists of clauses each of which can have one of two forms:

1. $H \leftarrow B_1, \ldots, B_n$, $(n \geq 0)$, i.e., *definite clauses*. Each B_i is of the form $p(\bar{t})$ where p is a predicate and $\bar{t}$ is a sequence of terms. In general, some of the $B_i s$ could be constraints as in CLP or CDB [4, 14–16].
2. $H \rightarrow C_1, \ldots, C_l \parallel B_1, \ldots, B_m$, $(l, m \geq 0)$, i.e., *optimization clauses*. $C_1, \ldots, C_l$ are constraints as in CLP that must be satisfied for this clause to be applicable to a goal; they must be read as antecedents of the implication. The variables that appear *only* on the RHS of the $\rightarrow$ clause are existentially quantified. The intended meaning of this clause is that the set of solutions to the head is some subset of the set of solutions to the body.

Moreover, the predicate symbols can be partitioned into three disjoint sets depending on the kinds of clauses used to define them:

- *C-predicates* appear only in the heads of definite clauses and the bodies of these clauses contain only *C-predicates* (*C* stands for *core*).
- *O-predicates* appear in the heads of only optimization clauses (*O* stands for *optimization*). For each ground instance of an optimization clause, the instance of the *O-predicate* at the head is the candidate for the optimal solution provided the corresponding instance of the body of the clause is true. The constraints that appear before the $\parallel$ in the body of

an optimization clause are referred to as the *guard* and must be satisfied in order for the head H to be reduced.

- *D-predicates* appear in the heads of only *definite* clauses and any one goal in the body of any of these clauses is either an *O-predicate* or a *D-predicate* (*D* stands for *derived from O-predicates*).

The **arbiter** part of a preference logic program, which specifies the **optimization criterion** for the *O-predicates*, has clauses of the form:

$$p(\bar{t}) \preceq p(\bar{u}) \leftarrow L_1, \ldots, L_n \quad (n \geq 0)$$

where p is an *O-predicate* and each L_i is an atom whose head is a *C-predicate* or a constraint as in CLP. In essence this form of the arbiter states that $p(\bar{t})$ *is less preferred than* $p(\bar{u})$ if $L_1, \ldots, L_n$.

A preference logic program is a triple of the form $(\mathcal{T}_C, \mathcal{T}_O, \mathcal{A})$ where $\mathcal{T}_C$ is made up of the definition of the *C-predicates*, $\mathcal{T}_O$ consists of the definitions of the *O-predicates* and *D-predicates*, and $\mathcal{A}$ consists of the arbiter clauses in the program.

The pre-interpretation I of interest to preference logic programs interprets functions such as $+$ over the appropriate domain (as in CLP) and leaves all other function symbols uninterpreted (as in Herbrand interpretations).

Theorem 1 ([10]). *For every program $P = (\mathcal{T}_C, \mathcal{T}_O, \mathcal{A})$, with n levels of O-predicates, its intended preference model at every level i, $1 \leq i \leq n$, exists and is unique up to an isomorphism.*

9.2. *Performing negotiation by constraints optimization*

Consider the following preference program of a quality manager:

$$P = (\mathcal{T}_C, \mathcal{T}_O, \mathcal{A})$$

with

$$\begin{aligned}\mathcal{T}_C = \{&\text{reliability('very high')} \leftarrow \text{throughput(X)},\\ &\qquad\qquad \text{X} < 100, \text{X} > 66,\\ &\text{reliability('high')} \leftarrow \text{throughput(X)}, \text{X} < 66\}\\ \mathcal{T}_O = \{&\text{u–reliability(X)} \rightarrow \text{reliability(X)}\}\end{aligned}$$

and

$$\begin{aligned}\mathcal{A} = \{&\text{u–reliability('medium')} \preceq \text{u–reliability('high')} \leftarrow .,\\ &\text{u–reliability('high')} \preceq \text{u–reliability('very high')} \leftarrow .,\\ &\text{u–reliability('medium')} \preceq \text{u–reliability('very high')} \leftarrow .\}\end{aligned}$$

throughput in $\mathcal{T}_C$ is an external function that we suppose implemented as a method of the object denoting the network.

Consider a user query $Q \| S$ whose view part S contains the constraint:

?- u–reliability('medium')

Depending on the current state of the whole system, our objective is to find the best qualifier for the parameter u–reliability. Suppose that the first rule of $\mathcal{T}_C$ triggers for some X between 66 and 100. Then, the fact u–reliability('very high') is generated. Now, the application of the rules of the arbiter part of our program can be applied to select the best solution. Hence, given the two facts u–reliability('medium') and u–reliability('very high'), the arbiter rule u–reliability('medium') $\preceq$ u–reliability('very high')$\leftarrow$. selects u–reliability('very high') as the best solution.

The optimization rules are useful mainly when several qualifiers are computed for a given parameter. In this case, they play a role of a selector.

We also consider the relaxation of preferences. Consider the following preference program:

$$\mathcal{T}_C = \{\text{reliability('very high', Y)} \leftarrow \text{throughput(Y)}, \text{Y} < 100, \text{Y} > 66,$$
$$\text{reliability('high', Y)} \leftarrow \text{throughput(Y)}, \text{Y} < 66\}$$
$$\mathcal{T}_O = \{\text{u–reliability(X, Y)} \rightarrow \text{reliability(X, Y)}\}$$

Now consider the query

?- u–reliability(X, Y), Y $>$ 50

Suppose the throughput at an instant is less than 50. Then, in this case, the above query fails. However, it is natural to want to compute an approximate solution without re-coding the defining of reliability. In the proposed framework, this requirement can be stated as follows:

?- RELAX u–reliability(X, Y)WRT Y $>$ 50

The above query works by widening the solution space for Y.

Rules can be written so that they allow negotiation in both directions. That is, they can express the fact that when resources are scarce, the resource demand must be degraded. On the contrary, the resource demand needs to be increased later on when resources are abundant.

This process of computing the best qualifier for a quality parameter is done for each parameter involved in the view part of a user-query. Let S be the user-specified quality part and S' the one computed by the preference program. To evaluate the corresponding

query over the database, we need to check whether S is subsumed by S'. This is because we assumed that S is the minimum QoS required by the user.

9.3. Quality subsumption

Let us formally define quality subsumption. For that we need additional definitions:

Definition 3 (Concrete Domains). A concrete domain $\mathcal{D} = (\mathsf{dom}(\mathcal{D}), \mathsf{pred}(\mathcal{D}))$ consists of:

- the domain $\mathsf{dom}(\mathcal{D})$,
- a set of predicate symbols $\mathsf{pred}(\mathcal{D})$, where each predicate symbol $P \in \mathsf{pred}(\mathcal{D})$ is associated with an arity n and an n-ary relation $P^{\mathcal{D}} \subseteq \mathsf{dom}(\mathcal{D})^n$,

An example of a concrete domain is the set of (nonnegative) integers with comparisons $(=, <, \leq, \geq, >)$.

We propose a framework for representing hierarchically structured quality parameters. A hierarchically structured quality parameter is a strict order set of elements.

Definition 4 (Quality Parameter). A *Quality Parameter* is a chain $(\mathcal{P}, \succ)$, where $\succ$ is a strict ordering.

Example. Let us consider the *reliability* quality parameter. The possible qualifiers for this parameter are Very High, High, Medium, Low and Very Low. We have the following total ordering between these elements: Very High $\succ$ High $\succ$ Medium $\succ$ Low $\succ$ Very Low.

In what follows, we consider each quality parameter P_q is associated with a unary predicate P, where $P(x)$ holds for all elements x belonging to the ordered set defined by P_q.

Definition 5 (Ordering on Quality Parameter Predicates). Let P_q be a quality parameter predicate and x and y be two elements of the chain defined by P_q. $P(x) \succeq P(y)$ if and only if $x \succ y$ in the chain associated with P_q.

We define a substitution as a mapping from variables to variables and constants, which is extended to be the identity on constants and generalized to free tuples in a natural fashion.

We extend substitution to abstract elements of chains as follows: let c and c' be two constants of a given chain. Then c could substitute c' if and only if $c \succ c'$. The extended substitution yields a generalized mapping.

Definition 6 (Quality Subsumption). Let S and S' be two quality specifications. We say that S' subsumes S (and we write $S \subseteq S'$) if there is a generalized mapping from S' to S.

9.4. *Algorithm for query evaluation*

In this section, we give an algorithm for query evaluation in video databases in the presence of resource fluctuations.

Algorithm 1 ($\mathcal{QE}$)

Require: a query $Q \| S$ (Q is the content part and S is the quality part)
// S is the minimum QoS required by th user
Ensure: evaluate Q by maintaining S satisfiable.

1: $DisplayedSet \leftarrow \emptyset$
// *computeQoS* Computes new quality parameters S' from S and the system state
// This means performing negotiation which consists in
// computing the new qualifiers for the parameters in S
// *computeQoS* stands for the preference program of a quality manager
2: *computeQoS*(S', S)
3: **if** $S' \succeq S$ **then**
4: Have the database to output the first answer x to Q
5: $DisplayedSet \leftarrow DisplayedSet \cup \{x\}$
// Rewrite Q to exclude the already displayed video sequences
6: Rewrite Q as $Q - DisplayedSet$
7: Repeat 2 until no answer satisfies Q
8: **end if**

10. Implementation

Our experimental prototype has been implemented on a Sun platform running SunOS 5.6. The prototype consists of several cooperating modules. We have implemented the user interface module using the QT[2] 2.00 C++ GUI application framework. Currently, the interface (see figure 6) allows users to specify queries on video objects with varying quality of service parameters. The QoS parameters that we have considered are those given in Table 2 as well as video resolution, audio quality and video/audio synchronization. The user's preferences are translated into a QoS specification that is then processed by the Quality Manager module (written in SWI-Prolog[3] [32]). This module also offers a user profile management component that can be used by a user to create/modify permanent quality profiles that should be associated with all his/her queries. Also, this component can be used by a DBA to assign quality profiles to users or to video documents (e.g., to associate a fixed quality of service to a document that is to be used by all users accessing that document).

The quality manager is organized as sketched in figure 5. It is implemented as a meta-interpreter written in SWI-Prolog. SWI-Prolog is a Prolog implementation based on a subset of the WAM (Warren Abstract Machine). It was developed as an open Prolog environment, providing a powerful and bi-directional interface to C. The meta-interpreter takes as input the query specified by the user, and three different programs, namely quality program, mapping program, and relaxation program.

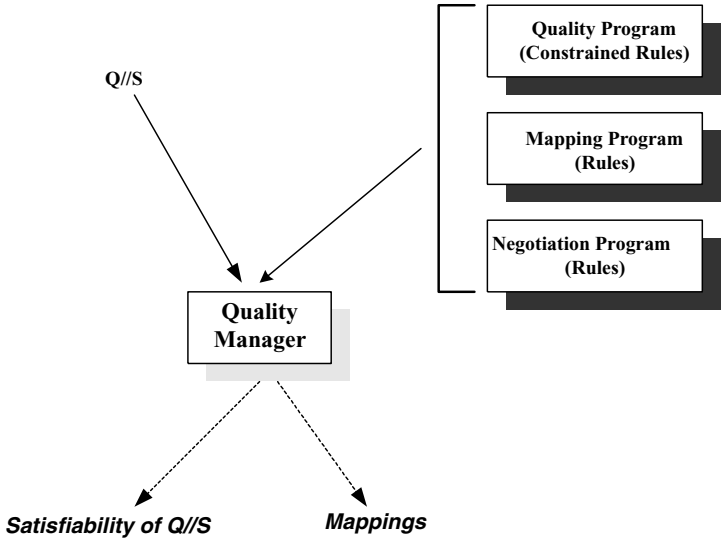


Figure 5. Inputs and outputs of the quality manager.

Figure 6. The graphical user interface (GUI).

The interface module is being extended to include an interactive component allowing users to dynamically relax/tighten their quality parameters using information provided by the Quality Manager module as well as information regarding various low level static (Memory size, Network bandwidth, etc.) or dynamic (available buffering space, CPU utilization, etc.) parameters.

When the Quality Manager module is able to satisfy the requirements of a query, the retrieved video data is output using our video rendering module. In the current status of our work, this module is derived from Berkeley's MPEG player. However, we are currently considering to rewrite this module in order to:

- fully support the wide range of quality parameters of users that are not offered by Berkeley's player,
- support other video/audio formats than the (too restrictive) ones allowed by Berkeley's player,
- dynamically adapt to performance fluctuations and/or user's on-line changing of quality parameters (Berkeley's player assumes a locally stored video documents).

11. Conclusion

There is a growing interest in video databases. As video libraries proliferate, aids to browsing and filtering become increasingly important tools for managing such exponentially growing information resources and for dealing with access problems. One of the central problems in the development of robust and scalable systems for manipulating video information[4] lies in supporting *quality of service*. We believe that formal settings will help understanding related problems. This will lead to the development of intelligent systems in order to effectively disseminate, retrieve, correlate and visualize video information.

This paper has described a logical framework for QoS specification in video databases. The primary contribution of this framework is that it allows some kinds of reasoning about QoS specifications.

The formalism used to specify quality constraints is also used to describe the mapping between parameters of different layers in the system, and to ensure the negotiation.

There are many interesting directions to pursue:

- An important direction is to extend our framework such that it can accommodate synchronization, concurrency and communication. By considering these aspects at an abstract logical level, it can be possible to predict and check the behavior of the system by reasoning and simulation based on specification, and to give sound reference basis for testing the implementation. This extension will be based on concurrency theory and distributed temporal logics [6].
- While some quality parameters are extensively investigated for low-level layers (i.e., network and operating system), this is not the case for the database level (storage, transaction, etc.). We believe that the support of QoS at the database level requires to devise new query evaluation and optimization strategies.
- Adaptive QoS management may enable a Video DBMS to overcome resource fluctuations by adaptations of media qualities. These adaptations should be optimized towards an

efficient utilization of available resources. More specifically, they should yield media objects and composite multimedia presentations with the best possible quality for the given resource availability. Accordingly, the best fitting adaptation is to be computed among the potentially large number of possible adaptations. This is because multimedia objects generally offer multiple parameters that may be adapted. Each of these parameters may have a large number of potential values. For adaptations of composite multimedia presentations, the combinatorial explosion problem is even larger. So, what are the algorithmical solutions for adaptation processing that has the general goal to compute corrective action sets to achieve the optimal adaptation?

We believe that quality of service specification and enforcement is an important area of research, and have laid the foundations for further research.

Notes

1. Quality of Presentation.
2. QT is a multi-platform C++ application framework that lets developers write single-source applications that run-natively-on Windows, Linux, Unix, Mac OS X and embedded Linux.
3. http://www.swi.psy.uva.nl/projects/SWI-Prolog/.
4. Multimedia information in general.

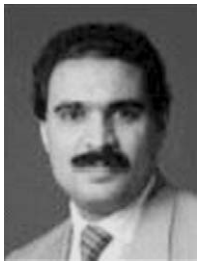
References

1. B. Bhargava, "Quality of Service in multimedia networks," Multimedia Tools and Applications (Special Issue), Vol. 17, Nos. 2/3, 2002.
2. T. Bolognesi and E. Brinksma, "Introduction to the ISO specification language LOTOS," Computer Networks and ISDN Systems, North-Holland, Vol. 14, No. 1, 1988.
3. A. Brown, S. Mantha, and T. Wakayama, "Logical reconstruction of constraint relaxation hierarchies in logic programming," in Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems (ISMIS'93), Trondheim, Norway LNAI 689, 1993, pp. 362–374.
4. J. Cohen, "Constraint logic programming languages," Communications of the ACM, Vol. 33, No. 7, pp. 52–68, 1990.
5. M.F. Daneshmand, R.R. Roy, and C.G. Savolaine, "Framework and requirements of quality of service for multimedia applications," in Proceedings of the 1997 IASTED International Conference on Intelligent Information Systems (IIS'97), Grand Bahama Island, Bahamas, 1997, pp. 466–474.
6. H.-D. Ehrich, C. Caleiro, A. Sernadas, and G. Denker, "Logics for secifying concurrent information systems," in Logics for Databases and Information Systems, J. Chomicki and G. Saake (Eds.), Kluwer Academic Publishers, 1998, pp. 167–198.
7. D. Ferrari, "Real-time communication in an internet-work," Journal of High Speed Networks, Vol. 1, No. 1, pp. 79–103, 1992.
8. S. Fisher and R. Keller, "Quality of service mapping in distributed multimedia systems," in Proceedings of the IEEE International Conference on Multimedia Networking (MMNet'95), M. Ikeda, S. Saito, and B. Sarikaya (Eds.), Aizu, Japan, 1995, pp. 132–141.
9. S. Gibbs, C. Breiteneder, and D. Tsichritzis, "Data modeling of time-based media," in Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data (SIGMOD'94), ACM Press: Minneapolis, Minnesota, USA, May 1994, pp. 91–102.
10. K. Govindarajan, "Optimization and relaxation in logic languages," Ph.D. thesis, Department of Computer Science, SUNY-Buffalo, 1997.

11. K. Govindarajan, B. Jayaraman, and S. Mantha, "Preference logic programming," in Proceedings of the Twelfth International Conference on Logic Programming (ICLP'95), MIT Press: Tokyo, Japan, 1995, pp. 731–745.
12. K. Govindarajan, B. Jayaraman, and S. Mantha, "Optimization and relaxation in constraint logic language," in Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'96), 1996, pp. 91–103.
13. A. Hampapur and R. Jain, "Video data management systems: Metadata and architecture," in Multimedia Data Management, A. Sheth and W. Klas (Eds.), Mc Graw Hill, 1998, pp. 245–286.
14. P. Van Hentenryck and V. Saraswat, "Strategic directions in constraint programming," ACM Comput. Surv., Vol. 28, No. 4, pp. 701–726, 1996.
15. J. Jaffar and J.-L. Lassez, "Constraint logic programming," in Proceedings of the Fourteenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'87), 1987, pp. 111–119.
16. P. Kanellakis, G. Kuper, and P. Revesz, "Constraint query languages," Journal of Computer and System Sciences (JCSS), Vol. 51, No. 1, pp. 26–52, 1995.
17. K.-W. Kim and K. Nahrstedt, "QoS translation and admission control for MPEG video," in Proceedings of the 5th IFIP International Workshop on Quality of Service (IWQOS'97), New York, 1997.
18. H. Knoche and H. de Meer, "Quantitative QoS mapping: A unifying approach," in Proceedings of the 5th IFIP International Workshop on Quality of Service (IWQOS'97), New York, 1997, pp. 347–358.
19. A. Lakas, G. Blair, and A. Chetwynd, "A formal approach to the design of QoS parameters in multimedia systems," in Proceedings of the 4th International Workshop on Quality of Service, Paris, France, 1996.
20. T. Lee, L. Sheng, T. Bozkaya, N.H. Balkir, Z.M. Özsoyoglu, and G. Özsoyoglu, "Querying multimedia presentations based on content," IEEE Transactions on Knowledge and Data Engineering (TKDE), Vol. 11, No. 3, pp. 361–385, 1999.
21. J. Lenstra, A. Rinnooy, and P. Brucker, "Complexity of machine scheduling problems," Annals of Discrete Mathematics, Vol. 1, 1977.
22. W. Li, S. Gauch, J. Gauch, and K.M. Pua, "VISION: A digital video library," in Proceedings of the 1st ACM International Conference on Digital Libraries, Bethesda, Maryland, USA, 1996, pp. 19–27.
23. D. Maier, J. Walpole, and R. Staehli, "Storage system architectures for continuous media data," in Proceedings of 4th International Conference on Foundations of Data Organization and Algorithms (FODO'93), Chicago, Illinois, USA, Vol. 730 of LNAI, Springer Verlag, 1993, pp. 1–18.
24. K.L. Myers, "Hybrid reasoning using universal attachment," Artificial Intelligence, No. 67, pp. 329–375, 1994.
25. K. Nahrstedt and R. Steinmetz, "Resource management in networked multimedia systems," IEEE Computer, Vol. 28, No. 5, pp. 52–63, 1995.
26. R. Needham and A. Nakamura, "Approach to real-time scheduling but is it really a problem for multimedia," in Proceedings of the Conference Network and Operating System Support for Digital Audio and Video (NOSSDAV'92), LNCS 712, Nov. 1992, pp. 32–39.
27. J.M. Spivey, The *Z* Notation: A Reference Manual. 2nd edn. Prentice-Hall International, 1992.
28. R. Staehli, J. Walpole, and D. Maier, "Quality of Service specification for multimedia presentations." ACM Multimedia Systems, Vol. 3, Nos. 5/6, pp. 251–263, 1995.
29. R. Steinmetz, "Analyzing the multimedia operating system," IEEE Multimedia, Vol. 2, No. 1, pp. 68–84, 1995.
30. N. Venkatasubramanian and K. Nahrstedt, "An integrated metric for video QoS," in Proceedings of the International Conference on Multimedia, Seattle, WA, USA, 1997, pp. 371–380.
31. J. Walpole, C. Krasic, L. Liu, D. Maier, C. Pu, D. McNamee, and D. Steere, "Quality of service semantics for multimedia database systems," in Database Semantics: Semantic Issues in Multimedia Systems, R. Meersman, Z. Tari, and S. Stevens (Eds.), Kluwer Academic Publishers, Jan. 1999.
32. J. Wielmaker, SWI-Prolog 3.3, Reference Manual, 2000. Available at http://www.swi.psy.uva.nl/projects/SWI-Prolog.
33. M. Wilson and A. Borning, "Hierarchical constraint logic programming," Journal of Logic Programming, Vol. 16, pp. 277–318, 1993.
34. C.-H. Wu, R.J. Miller, and M.T. Liu, "Querying multimedia presentations," in Proceedings of the IEEE Conference on Protocols for Multimedia Systems—Multimedia Networking (PROMSMmNET'97), 1997, pp. 64–73.

35. A. Zhang and T.V. Johnson, "A framework for supporting quality-based multimedia presentation in educational digital libraries," in Proceedings of the Advanced Digital Libraries Forum, Washington, DC, USA, May 1997, pp. 102–113.
36. A. Zhang and S. Multani, "Implementation of video presentation in database systems," in Proceedings of Storage and Retrieval for Still Image and Video Databases IV, IS&T SPIE, San Jose, California, Vol. 2670, 1996, pp. 228–238.
37. W. Zhao and K. Ramamritham, "Simple and integrated heuristic algorithms for scheduling tasks with time and resource constraints," Journal of Systems and Software, Vol. 7, 1987.

Elisa Bertino is a Professor of Computer Science at the University of Milan. Her research interests include security, database systems, object technology, and multimedia systems. She recently co-authored the book Intelligent Database Systems (Addison-Wesley, 2001).



Ahmed K. Elmagarmid is a Professor of Computer Science at Purdue University. His research interests are in the areas of video databases, multidatabases, data quality and their application in telemedicine and digital government. He is the author of several books on databases and multimedia.

Mohand-Said Hacid is a Professor at the University Claude Bernard Lyon 1. His research interests include knowledge representation and reasoning, and data models and query languages for multimedia databases and semantic Web.