A Database Approach to Quality of Service Specification in Video Databases

Elisa Bertino[*] **Ahmed K. Elmagarmid**[†] **Mohand-Saïd Hacid**[†]

[*]University of Milano, Italy
bertino@dsi.unimi.it

[†]Purdue University, USA
{ake,mshacid}@cs.purdue.edu

Abstract

Quality of Service (QoS) is defined as a set of perceivable attributes expressed in a user-friendly language with parameters that may be subjective or objective. 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. 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.*

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, involve 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, computing the result of a query in a video[1] database is more a question of *quality* rather than correctness. 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.

We partition the QoS parameters into two subsets, namely application-dependent parameters and application-independent parameters. For example, most electronic commerce applications require multi-media presentation to the customer from the vendors, 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 interstream synchronization. Researchers have yet to determine the best set of QoS parameters for multimedia systems. Table 1 shows some

[1]More generally multimedia.

common QoS parameters in the multimedia community (mainly for video).

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* ...

Table 1: Examples of Possible Quality of Service Parameters

This paper advocates the use of a 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 [6], we consider queries as 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, that allows 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.

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.

Paper outline: This paper is organized as follows. Section 2 explains our view regarding the role of a database management system for managing quality of service. Section 3 describes the query language. The language allows to express search queries constrained by quality parameters. Section 4 develops an algorithm for deriving implicit constraints from explicit constraints that must be satisfied to guarantee the quality required by the user. Section 5 shows how the mapping between parameters of different layers in the system can be specified by simple rules. We conclude in Section 6 by anticipating on the necessary extensions.

2 Quality of Service and DBMS

The database system layer is responsible for interpreting the user's request (objects or documents requested and the quality of presentation[2] parameters), translating them into QoS parameters for the subsystems, and orchestrating the negotiation between the various components of the overall system. The output of this process is, for example, a translation of the quality of presentation parameters as well as the query execution plan for retrieving the data.

The objects that need to be retrieved in order to answer the query are identified by the database system using the constraints specified in the query as well as the quality of presentation parameters (such as resolution or frame rate). To support various levels of quality, the system may store *several versions* of some objects (possibly at different servers) or employ techniques such as wavelets to enable the generation of different *quality versions.* Thus depending upon the query and quality, the database may choose among a set of objects that need to be retrieved.

For example, consider a user request for a video clip V. Let this clip be stored at full resolution, V_1, and 50% resolution, V_2 at sites S_1 and S_2 respectively. Suppose that the user's quality requirements can be met with the 50% resolution version. The database can therefore retrieve V_2 from S_2, or retrieve V_1 from S_1 and apply a filter that reduces the resolution to 50%. Clearly, the second option involves an extra step (and cost) of filtering. However, if the operating system or the network at S_2 is overloaded this extra cost may be acceptable and allow timely delivery of the data.

The database needs to be able to identify alternative plans for the retrieval and delivery of the objects. Furthermore, it needs to evaluate the effectiveness of each plan in complying with the user's quality requirements. This task is akin to the process of query optimization in traditional databases wherein the database has to choose among a collection of alternative execution plans based upon their expected performance. There are several key differences between traditional query optimization and multimedia query optimization: (1) for multimedia, the database needs to first determine whether or not a plan will violate the user's requirements of quality – this involves QoS translation; (2) Furthermore, if none of the plans are acceptable, then perhaps the constraints (i.e., the user's requirements) can be relaxed – this involves QoS re-negotiation; and (3) whereas traditional optimization is based upon estimates of cost (heuristically or statistically determined), multimedia optimization needs to obtain current information about the cost – how many requests are currently being serviced, what is the available bandwidth now, where is the disk head currently, etc.

Our model, language, and the constraint database will enable these query optimization techniques. Using the query component of the user request and the content store, the

[2]Quality of presentation parameters are a subset of quality of service parameters.

DBMS identifies the data items that need to be retrieved. Note that alternative copies may be identified for a given request. The DBMS next generates alternative plans for executing the query (i.e. delivering the requested data to the user with the requested quality). The alternative plans are also expressed as constraints. These constraints specify the retrieval of data from disk or from the cache, the application of transforming filters, transmission of data between nodes, and so on. The DBMS evaluates the feasibility of the alternative plans by evaluating the constraints for the plan along with other constraints of the system. As mentioned earlier, for multimedia queries, it is important to determine the status of various parameters at query evaluation time. These are captured in terms of dynamic constraints that are evaluated when each query is tested for satisfiability.

If the DBMS finds a plan that is feasible, i.e., the system can execute the plan with its current state and meet the required level of quality there are two options: (i) it could accept this plan and execute it; or (ii) it could continue to look for other feasible plans. In the latter case, the DBMS can choose an optimal plan among several feasible plans based upon expected costs as with traditional query optimization. In this paper we do not consider such optimization. If no feasible plan is found, the query can be rejected. Alternatively, the quality constraints can be relaxed to allow plans that give a slightly lower quality. This is the process of QoS re-negotiation. Thus, the DBMS can try to obtain a feasible plan with high quality and progressively lower the quality until a feasible plan is found. If no plan is found that meets even the lowest acceptable level of quality for the user, then the query is relaxed.

3 Expressing Queries Involving QoS/QoP

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 640x480 pixel display window, but a quality specification that requires only 320x240 pixels of resolution. In this case the player may be able to avoid generating the full resolution images from a video sequence.

Definition 1 (Query) *A query is of the form:*

$$\mathsf{Q}||\mathsf{S}$$

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 1 *The following query:*

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

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.

4 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 1 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 meta database are fed to a module called evaluation and elaboration, which retrieves objects answers to the query and derive additional constraints. For each constrained object retrieved from the meta database, 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.

4.1 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

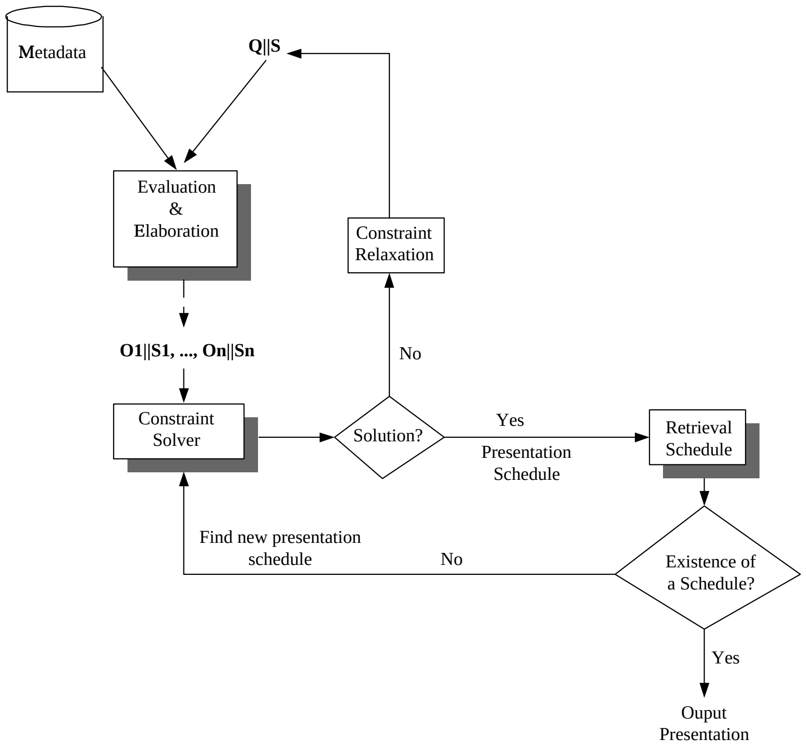

Figure 1: Interaction between the different modules

retrieved object O we build two sets $\mathcal{Q}.\mathcal{S}$ where $\mathcal{Q}$ is called the set of facts and $\mathcal{S}$ is called the set of constraints. $\mathcal{S}$ is the union of S and the constraints attached to O in the meta database.

An *elaboration rule* is an if-then rule that adds constraints to a set of constraints. Table 2 shows examples of elaboration rules

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:

$$\begin{array}{l}\mathsf{video}(X), \mathsf{audio}(Y), X \text{ contains } Z,\\ Z \text{ name } "Clinton", X \text{ has\_audio } Y||\\ \mathsf{display\_time}(t_d), t_d \leq 5, \mathsf{resolution}(X, 'high')\end{array}$$

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

$$\mathcal{Q} = \{\mathsf{video}(v), \mathsf{audio}(a), v \text{ contains } o, o \text{ name } "Clinton", v \text{ has\_audio } a\}$$

$$\mathcal{S} = \{\mathsf{display\_time}(t_d), t_d \leq 5, \mathsf{resolution}(v, 'high')\}$$

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

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

Now, by using the elaboration rule:

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

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

Let t_v be the display time that the video media can provide from $t_{v_{start}}$ and t_a be the display time the audio media can provide from $t_{a_{start}}$. Then, the constraints $t_{v_{start}} + t_d \leq t_{v_{start}} + t_v$ and $t_{a_{start}} + t_d \leq t_{a_{start}} + t_a$ 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 audio medium can provide from t_Y
t_X and t_Y being the starting time for display of X and Y, respectively

The augmented set of constraints is then:

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

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 [4]. Effective heuristic algorithms exist for this problem [9] which are sensitive to the uncertainty in task completion times.

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

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

$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{t_X + t_d \leq t_X + t_v, t_Y + t_d \leq t_Y + t_a\}$
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 starting time for display of X and Y, respectively

$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{t_d \leq c'\} \setminus \{t_d \leq c\}$
if $\mathcal{S}$ contains video(X)
and $\mathcal{S}$ contains $t_d \leq c$ and X *time* c'
and $c' \leq c$

$\mathcal{Q}.\mathcal{S} \rightarrow \mathcal{Q}.\mathcal{S} \cup \{t_{X_{end}} \leq t_{Y_{start}}\}$
if $\mathcal{S}$ contains before(X, Y)

Table 2: Examples of elaboration rules

5 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, [2]). 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, [7, 5]). The aim of buffer management is to avoid jitter and to minimize copy operations [7]. 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 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 3.
Consider the set of network parameters given in table 4. A possible mapping of the user's quality of service parameters of table 3 to network parameters of table 4 is depicted in figure 2.

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: Some User Quality of Service Parameters.

Parameter	Domain
Throughput	*bytes per second*
MTU (Maximum Transfer Unit)	*bytes*
reliability	*percent*
burstiness	*integer*
delay	*milliseconds*
jitter	*milliseconds*

Table 4: Examples of Transport QoS Parameters.

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

D_P ← S_P

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* [3]. 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

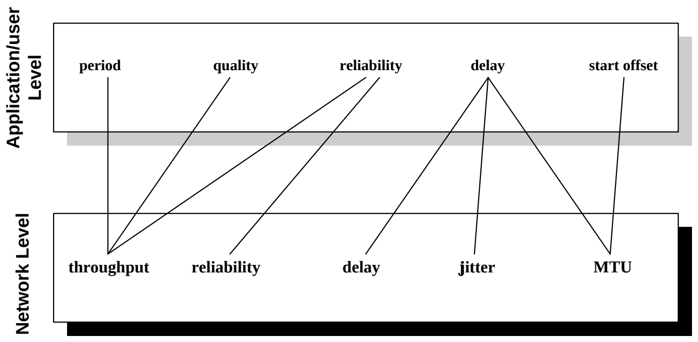

Figure 2: Mapping of Application QoS Parameters to Network QoS Parameters.

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:

```
throughput(redundancy→33, AdFEC_Type→'FEC_2_1') ←
                    reliability(qualifier→ X), X = '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).

6 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[3] lies in supporting *quality of service*. We believe that formal settings will help understanding related problems.

This paper has described a logical framework for QoS specification in video databases. The primary contributions 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 [1]
- 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.

[3]Multimedia information in general.

References

[1] Hans-Dieter Ehrich, Carlos Caleiro, Amilcar Sernadas, and Grit Denker. Logics for Secifying Concurrent Information Systems. In *Logics for Databases and Information Systems, Jan Chomicki and Gunter Saake Eds.*, pages 167–198. Kluwer Academic Publishers, 1998.

[2] D. Ferrari. Real-Time Communication in an Internet-Work. *Journal of High Speed Networks*, 1(1):79–103, 1992.

[3] S. Fisher and R. Keller. Quality of Service Mapping in Distributed Multimedia Systems. In *Proceedings of the IEEE International Conference on Multimedia Networking (MM-Net'95), Aizu, Japan, M. Ikeda, S. Saito, B. Sarikaya (eds.)*, pages 132–141, 1995.

[4] J. Lenstra, A. Rinnooy, and P. Brucker. Complexity of Machine Scheduling Problems. *Annals of Discrete Mathematics*, 1, 1977.

[5] 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*, pages 32–39, November 1992.

[6] Richard Staehli, Jonathan Walpole, and David Maier. Quality of Service Specification for Multimedia Presentations. *ACM Multimedia Systems*, 3(5/6):251–263, November 1995.

[7] R. Steinmetz. Analyzing the Multimedia Operating System. *IEEE Multimedia*, 2(1):68–84, 1995.

[8] Jan Wielmaker. SWI-Prolog 3.3, Reference Manual, http://www.swi.psy.uva.nl/projects/SWI-Prolog. 2000.

[9] W. Zhao and K. Ramamritham. Simple and Integrated Heuristic Algorithms for Scheduling Tasks with Time and Resource Constraints. *Journal of Systems and Software*, 7, 1987.